Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розроблення системи автоматизації логістичного центру виробничого підприємства
(тема)

Виконав:
здобувач 2 року навчання,
групи КТРСм-24-1
Інна ТКАЧЕНКО
(власне ім'я, прізвище)

Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Комп'ютеризовані та робототехнічні системи
(повна назва освітньої програми)

Керівник доц. Кирило ХРУСТАЛЬОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Я, Ткаченко Інна Андріївна, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавала і не одержувала недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовувала штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

01 грудня 2025 р.  Інна ТКАЧЕНКО

Харківський національний університет радіоелектроніки

Факультет АКТ

Кафедра КІТАР

Рівень вищої освіти другий (магістерський)

Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Комп'ютеризовані та робототехнічні системи
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри КІТАР_____
(підпис)

«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ткаченко Інні Андріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення системи автоматизації логістичного центру виробничого підприємства
затверджена наказом університету від 10 листопада 2025 р. № 1018 Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 23 грудня 2025 р.
3. Вихідні дані до роботи
3.1 Система призначена для автоматизації складських процесів логістичного центру виробничого підприємства на базі Odoo.
3.2 Вхідні дані: номенклатура та залишки товарів; документи складських операцій; дані сканування з ТЗД (штрих-код/RFID).
3.3 Метод обробки даних: транзакційна обробка складських операцій у WMS з контролем відповідності планових і фактичних параметрів та фіксацією розбіжностей (Divergence).
3.4 Середовище реалізації: Odoo 14.0 CE, PostgreSQL, Python (backend), OWL/JS та XML (frontend/конфігурація), інструменти розробки PyCharm, pgAdmin 4.
3.5 Вимоги до результатів: забезпечення точного обліку запасів у реальному часі, зменшення ручних операцій і помилок.
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Проаналізувати сучасний стан управління складською логістикою.
4.2 Ознайомитися з існуючими WMS-рішеннями та обґрунтувати вибір платформи розробки.
4.3 Розробити структуру та архітектуру системи.
4.4 Розробити алгоритм роботи програмного забезпечення складської підсистеми.
4.5 Реалізувати та протестувати програмний модуль WMS на базі Odoo.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
Графічний матеріал у вигляді презентації – 22 арк. ф. А4

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	*Аналіз сучасного стану управління складською логістикою*	25.09.2025	*виконано*
2	*Аналіз існуючих WMS-рішень та обґрунтування вибору платформи розробки*	15.10.2025	*виконано*
3	*Розроблення структурної схеми та архітектури системи автоматизації логістичного центру підприємтсва*	10.11.2025	*виконано*
4	*Розроблення алгоритму роботи програмного забезпечення складської підсистеми*	25.11.2025	*виконано*
5	*Реалізація та тестування програмного модуля WMS на базі Odoo*	05.12.2025	*виконано*
6	*Оформлення пояснювальної записки*	15.12.2025	*виконано*
7	*Подання роботи на нормоконтроль*	21.12.2025	*виконано*
8	*Подання роботи для перевірки роботи на академічну доброчесність*	21.12.2025	*виконано*
9	*Подання роботи на рецензію*	22.12.2025	*виконано*
10	*Подання роботи на підпис зав. кафедри*	22.12.2025	*виконано*
11	*Подання кваліфікаційної роботи в ЕК*	23.12.2025	

Дата видачі завдання 01 вересня 2025 р.

Здобувач _____ Інна ТКАЧЕНКО
(підпис)

Керівник роботи _____ доц. Кирило ХРУСТАЛЬОВ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 128 с., 2 табл., 46 рис., 2 дод., 19 джерел посилання.

АВТОМАТИЗАЦІЯ, СКЛАДСЬКА ЛОГІСТИКА, ERP, WMS, RFID, RASPBERRY PI, СИСТЕМА УПРАВЛІННЯ, ОПТИМІЗАЦІЯ ПОТОКІВ.

Кваліфікаційна робота присвячена розробленню системи автоматизації логістичного центру виробничого підприємства на базі платформи ERP Odoo у вигляді підсистеми класу Warehouse Management System (WMS).

Метою кваліфікаційної роботи є підвищення ефективності керування складськими процесами за рахунок створення системи автоматизації, яка забезпечує раціональну організацію потоків матеріальних ресурсів, мінімізацію впливу людського фактора та підвищення точності виконання логістичних операцій.

Об’єкт дослідження – процес управління функціонуванням складської системи, що включає логістичні операції виробничо-логістичного комплексу.

Предмет дослідження – програмний комплекс автоматизованої системи управління складом, яке реалізує взаємодію між користувачем, технічними засобами та інформаційними потоками з метою оптимізації прийняття управлінських рішень і підвищення продуктивності складських процесів.

У роботі проаналізовано сучасні підходи до управління складськими процесами та обґрунтовано актуальність впровадження WMS для підвищення ефективності складської логістики підприємства. Проведено огляд і порівняння провідних програмних рішень для управління складом, зокрема комерційних систем (SAP EWM, Oracle Warehouse Management, Microsoft Dynamics 365 тощо) та open-source платформи Odoo. На підставі аналізу зроблено висновок про доцільність використання Odoo як базової технологічної платформи, що

пояснюється її гнучкістю, відкритим вихідним кодом та можливістю інтеграції з іншими підсистемами підприємства.

Розроблено проєктні рішення для створюваної системи: запропоновано трирівневу архітектуру (база даних – сервер бізнес-логіки – інтерфейс користувача), що забезпечує розподіл функцій та масштабованість системи, а також побудовано діаграму класів, яка моделює основні сутності WMS-підсистеми (складські документи, рух товарів, місця зберігання, номенклатура товарів, контрагенти тощо) та взаємозв’язки між ними. На основі цих схем визначено функціональність системи і розроблено алгоритм роботи програмного забезпечення, який описує послідовність операцій від моменту створення замовлення до завершення складського процесу з перевіркою та фіксацією можливих розбіжностей.

ABSTRACT

Explanatory note: 128 pages, 2 tables, 46 figures, 2 applications, 19 reference sources.

AUTOMATION, WAREHOUSE LOGISTICS, ERP, WMS, RFID, RASPBERRY PI, MANAGEMENT SYSTEM, FLOW OPTIMISATION.

The qualification work focuses on developing an automation system for the logistics center of a manufacturing enterprise, based on the Odoo ERP platform, in the form of a Warehouse Management System (WMS) subsystem.

The aim of the thesis is to enhance the efficiency of warehouse process management by developing an automation system that ensures the rational organization of material resource flows, minimizes the impact of the human factor, and improves the accuracy of logistics operations.

The object of the study is the process of managing the warehouse system's functioning, which encompasses the logistics operations of the production and logistics complex.

The subject of the study is a software complex for an automated warehouse management system, which implements interaction between the user, technical means, and information flows to optimize management decisions and increase the productivity of warehouse processes.

The paper analyses modern approaches to warehouse process management and substantiates the relevance of implementing WMS to improve the efficiency of warehouse logistics at the enterprise. A review and comparison of leading software solutions for warehouse management, in particular commercial systems (SAP EWM, Oracle Warehouse Management, Microsoft Dynamics 365, etc.) and the open-source platform Odoo, was conducted. Based on the analysis, it is concluded that using Odoo

as the basic technological platform is advisable due to its flexibility, open-source code, and the possibility of integration with other enterprise subsystems.

Design solutions for the system being created were developed: a three-level architecture (database – business logic server – user interface) was proposed, which ensures the distribution of functions and scalability of the system, and a class diagram was constructed that models the main entities of the WMS subsystem (warehouse documents, movement of goods, storage locations, nomenclature of goods, counterparties, etc.) and the relationships between them. Based on these diagrams, the system's functionality was determined, and a software algorithm was developed that describes the sequence of operations from the moment an order is created to the completion of the warehouse process, including verification and recording of any possible discrepancies.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АСУ – автоматизована система управління;

ІТ – інформаційні технології;

КІТАР – кафедра комп’ютерно-інтегрованих технологій, автоматизації та робототехніки;

ПЛК – програмований логічний контролер;

СКБ – студентське конструкторське бюро;

ТЗД – термінал збору даних;

ХНУРЕ – Харківський національний університет радіоелектроніки;

AI – Artificial Intelligence;

ERP – Enterprise Resource Planning;

HMI – Human-Machine Interface;

IoT – Internet of Things;

NFC – Near Field Communication;

RFID – Radio Frequency Identification;

SaaS – Software as a Service;

SCM – Supply Chain Management;

WMS – Warehouse Management System.

ВСТУП

Сучасні промислові підприємства функціонують в умовах зростання обсягів виробництва і товарообігу, що висуває підвищені вимоги до організації роботи складів і логістичних центрів. Ефективне управління складською логістикою є критично важливим для забезпечення безперебійності виробничих процесів та своєчасного постачання продукції споживачам. Традиційні методи керування складом (на основі паперової документації або розрізнених електронних таблиць) не відповідають сучасним вимогам ні за швидкодією обробки інформації, ні за точністю обліку. Ручне введення даних призводить до значних витрат і високого ризику помилок, що може спричиняти як надлишки, так і дефіцит запасів, втрати часу на пошук товарів та інші проблеми.

Таким чином, автоматизація складських процесів постає актуальним завданням, вирішення якого дозволяє підвищити продуктивність праці, мінімізувати операційні втрати та оптимізувати товаропотоки на підприємстві. Впровадження сучасної інформаційної системи управління складом забезпечує прозоре відстеження руху матеріальних ресурсів в режимі реального часу, своєчасне поповнення запасів і виконання замовлень, що в кінцевому рахунку сприяє підвищенню конкурентоспроможності підприємства.

Такий підхід відповідає цілям сталого розвитку ООН, зокрема Цілі 9 «Індустріалізація, інновації та інфраструктура» та Цілі 12 «Відповідальне споживання і виробництво», адже сприяє впровадженню інноваційних технологій та оптимізації використання ресурсів.

Метою кваліфікаційної роботи є підвищення ефективності керування складськими процесами за рахунок створення системи автоматизації, яка забезпечує раціональну організацію потоків матеріальних ресурсів, мінімізацію впливу людського фактора та підвищення точності виконання логістичних операцій.

Об’єкт дослідження – процес управління функціонуванням складської системи, що включає усі логістичні операції виробничо-логістичного комплексу.

Предмет дослідження – програмний комплекс автоматизованої системи управління складом, яке реалізує взаємодію між користувачем, технічними засобами та інформаційними потоками з метою оптимізації прийняття управлінських рішень і підвищення продуктивності складських процесів.

Для реалізації поставленої мети необхідно виконати наступні завдання:

– провести аналіз сучасного стану управління складською логістикою;

– провести аналіз існуючих WMS-рішень та обґрунтувати вибір платформи розробки;

– розробити структурну схему та архітектуру системи автоматизації логістичного центру виробничого підприємства;

– розробити алгоритм роботи програмного забезпечення складської підсистеми;

– реалізувати та провести тестування програмного модуля WMS на базі Odoo;

– оформити пояснювальну записку згідно з рекомендаціями [1-8], та вимогами ДСТУ 3008:2015 [9]. За результатами роботи було опубліковано тези доповіді, наведено їх текст у додатку А [10].

1 СУЧАСНИЙ СТАН ТА АКТУАЛЬНІСТЬ СИСТЕМ АВТОМАТИЗАЦІЇ СКЛАДСЬКОЇ ЛОГІСТИКИ

1.1 Базові концепції та класифікація систем управління підприємством і складом

1.1.1 Системи ERP

ERP (Enterprise Resource Planning) – це комплексна інформаційна платформа, призначена для узгодженого управління всіма ключовими бізнес-процесами організації. Вона об’єднує в єдине ціле фінансові, виробничі, логістичні, торговельні та адміністративні підсистеми, забезпечуючи їхню автоматизацію та взаємодію (рис. 1.1).

Рисунок 1.1 – Функціональні області ERP

Витоки ERP-рішень сягають 1960-х років, коли автоматизація бухгалтерського обліку лише починала розвиватися. Тодішні системи функціонували на великих обчислювальних машинах і включали базові модулі

для фінансів, виробництва та реалізації продукції. Протягом 1990-х років розвиток обчислювальних технологій і поширення клієнт-серверної архітектури значно розширили функціонал ERP. Системи набули масштабованості, інтегрованості та адаптивності. До їхнього складу почали входити підсистеми управління клієнтськими відносинами (CRM) та ланцюгом постачання (SCM).

Початок 2000-х ознаменувався інтеграцією ERP із вебтехнологіями та хмарними сервісами. Перехід до моделі SaaS (Software as a Service) дав можливість підприємствам впроваджувати ERP без великих капіталовкладень у серверне обладнання й технічну підтримку. У сучасних умовах ERP-платформи є основою цифрової інфраструктури підприємств різних масштабів і напрямів діяльності. Вони охоплюють усі ключові сфери управління – від фінансових потоків і ресурсів до логістики, продажів та операційного планування [10].

Центральним елементом ERP є єдина база даних, що об’єднує інформаційні потоки всіх підрозділів компанії. Така інтеграція забезпечує безперервний обмін даними, підвищує точність аналітики, скорочує час на обробку інформації та підтримує обґрунтованість управлінських рішень.

Ключовим принципом функціонування ERP є концентрація даних у спільному інформаційному середовищі. Це дозволяє всім учасникам бізнес-процесів – від керівництва до виконавців – працювати з однаковими, актуальними й узгодженими даними. На відміну від практики використання розрізнених таблиць чи баз даних для обліку запасів, замовлень і фінансів, ERP об’єднує всі ці елементи в єдину, логічно узгоджену систему інформаційних потоків.

Структура ERP має модульну будову: кожен компонент відповідає за певний напрям діяльності підприємства. Зокрема, фінансовий модуль охоплює бухгалтерський облік, управління розрахунками, аналіз витрат і планування бюджету, тоді як модуль продажів забезпечує контроль за клієнтами, замовленнями та складськими запасами.

Важливою рисою ERP є її адаптивність: компанія може розгорнути лише потрібні модулі й поступово розширювати функціонал у міру зростання.

Система підтримує інтеграцію з іншими програмними рішеннями, що сприяє ефективній взаємодії між усіма структурними підрозділами. Незалежно від масштабу організації, ERP виступає універсальним інструментом для автоматизації бізнесу, сприяє підвищенню ефективності операцій, скороченню витрат і зростанню продуктивності праці. Використання ERP підвищує точність прогнозування фінансових показників, дозволяє ефективно управляти запасами, покращує координацію між департаментами та скорочує часові витрати на ухвалення управлінських рішень. Завдяки централізованій базі ERP забезпечує швидкий доступ до достовірних даних, що значно спрощує аналітику та управління. Автоматизація більшості операцій знижує ризик людських помилок і підвищує ефективність праці персоналу.

В умовах стрімкого збільшення обсягів корпоративних даних роль ERP-систем лише зростає. Вони гарантують своєчасний доступ до актуальної інформації, підтримують оперативність управління й зміцнюють конкурентні позиції організацій на ринку.

1.1.2 WMS системи

WMS (Warehouse Management System) – це спеціалізоване програмне забезпечення, створене для автоматизації та вдосконалення процесів, пов’язаних із керуванням складськими операціями. Воно забезпечує контроль і відстеження всіх етапів роботи складу – від приймання та розміщення товарів до їх підготовки до відвантаження й доставки клієнтам. Використання таких систем підвищує ефективність логістичних процесів, точність виконання операцій і дозволяє раціональніше використовувати складські площі, збільшуючи продуктивність на від 10 % до 30 %.

Формування концепції систем управління складом тривало десятиліттями й проходило кілька ключових етапів. Перші WMS-рішення мали примітивний характер і передбачали ручне введення даних, однак із розвитком обчислювальної техніки та появою засобів автоматизації вони поступово еволюціонували у складні інформаційні системи [11].

Із розширенням логістичних операцій і зростанням складності бізнес-процесів WMS отримали нові можливості (рис. 1.2):

– управління запасами;

– контроль за постачаннями;

– оптимізацію розміщення товарів;

– прогнозування попиту.

Рисунок 1.2 – Процеси управління складом

Подальший розвиток став можливим завдяки інтеграції WMS із системами ERP. Це дало змогу забезпечити узгоджену роботу різних підрозділів підприємства, об’єднавши фінансові, виробничі й логістичні процеси в єдину інформаційну структуру.

Використання технологій Інтернету речей (IoT) відкрило новий етап у розвитку WMS. Підключення сенсорів і RFID-міток дозволяє в реальному часі відстежувати рух товарів, контролювати умови зберігання й автоматизувати низку складських операцій.

Сучасною тенденцією є застосування штучного інтелекту та алгоритмів машинного навчання, які сприяють побудові адаптивних моделей управління запасами. Такі системи здатні прогнозувати попит, визначати оптимальні місця зберігання й підтримувати прийняття ефективних управлінських рішень.

Отже, еволюція WMS від ручних процесів до комплексних цифрових платформ демонструє перехід від елементарних систем обліку до інтелектуальних рішень, що базуються на сучасних технологіях та аналітиці даних [12].

Типова WMS система виконує такі функції:

– моніторинг запасів у реальному часі;

– планування та оптимізація розміщення товарів;

– автоматизація процесів комплектації й відправлення замовлень;

– проведення інвентаризації та аудит складських залишків;

– контроль ефективності роботи персоналу.

1.1.3 Термінал збору даних

Термінали збору даних (ТЗД) – це ключові інструменти у складі сучасних систем управління складом, які забезпечують підвищення точності та швидкості виконання операцій.

Такі пристрої являють собою портативні модулі, оснащені сканерами штрих-кодів, дисплеєм і клавіатурою, що дозволяє працівникам у режимі реального часу реєструвати, обробляти й передавати інформацію безпосередньо під час виконання логістичних завдань. Завдяки цьому досягається постійна синхронізація між фізичними товарними позиціями та їх цифровими аналогами в базі даних системи WMS.

Використання ТЗД надає можливість виконувати широкий спектр операцій:

– сканування штрих-кодів та фіксацію місцезнаходження товарів;

– контроль кількості та руху продукції на складі;

– проведення інвентаризації та переміщення запасів;

– оформлення приймання й відвантаження замовлень.

Уся зібрана інформація миттєво передається до WMS, що забезпечує актуальність даних і дозволяє оперативно оновлювати статус кожної операції.

Застосування ТЗД у системах управління складом істотно підвищує ефективність логістичних процесів, мінімізує ймовірність помилок, характерних для ручного введення даних, та скорочує час на обробку інформації. Працівникам достатньо лише відсканувати код товару – усі подальші дії виконуються автоматично.

Крім основних функцій, сучасні моделі терміналів підтримують розширені можливості:

– бездротове з’єднання для швидкої передачі даних;

– розпізнавання голосових команд;

– роботу в умовах низького освітлення чи екстремальних температур.

Таким чином, використання ТЗД є невід’ємною складовою ефективного функціонування складських систем. Вони створюють зв’язок між фізичними об’єктами та цифровими даними, забезпечуючи точність обліку, оперативність оновлення інформації та повну автоматизацію процесів управління складом [13].

1.2 Огляд комерційних ERP систем та їх порівняльна характеристика

1.2.1 ERP SAP

SAP – одна з найвідоміших інтегрованих ERP-платформ, що пропонує комплексні рішення для ведення складського обліку та управління ресурсами підприємства. Система охоплює широкий спектр бізнес-функцій, зокрема:

– управління запасами;

– прогнозування попиту;

– планування логістичних і виробничих процесів;

– фінансовий облік і аналітику.

SAP ERP підтримує багатомовний інтерфейс (рис. 1.3) і роботу з різними валютами, що робить її зручною для міжнародних корпорацій. Попри високу

функціональність, система вважається складною у впровадженні та обслуговуванні, а її вартість залишається однією з головних перешкод для невеликих компаній.

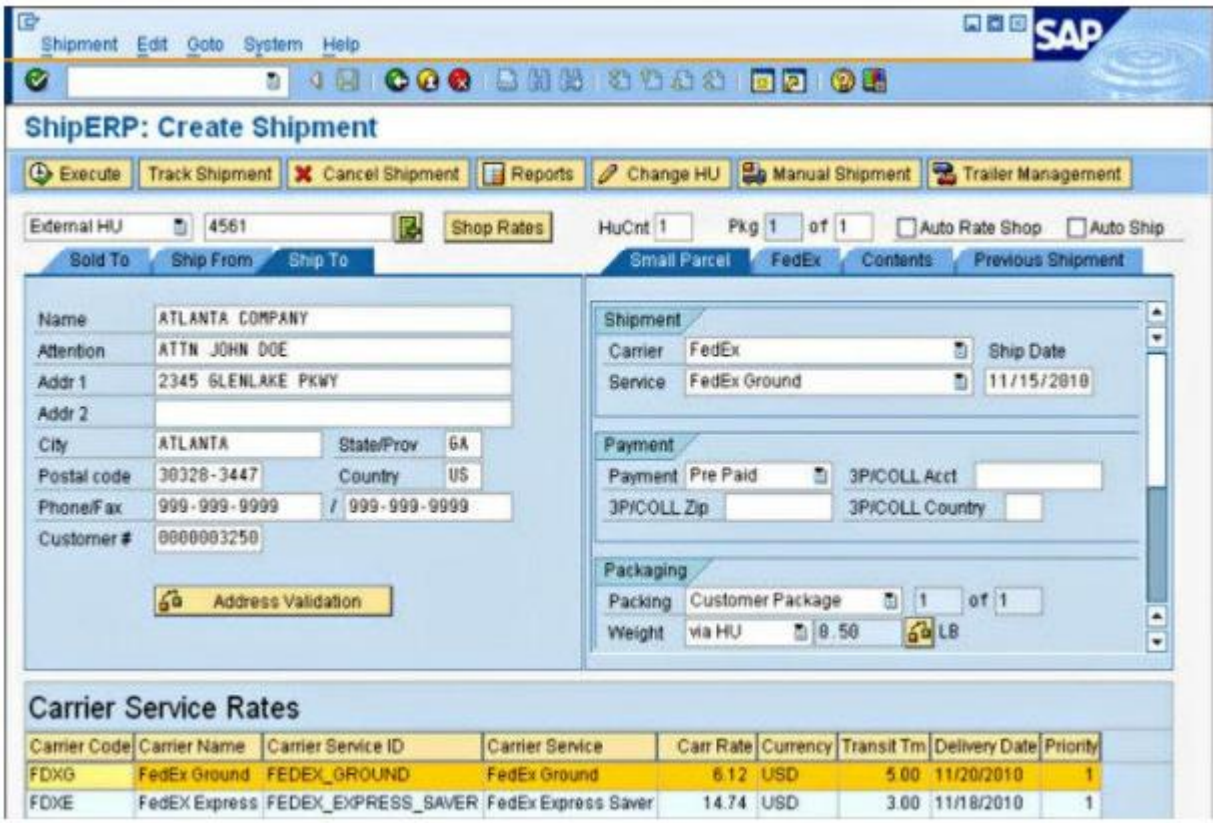


Рисунок 1.3 – Інтерфейс SAP ERP

Разом з тим, SAP ERP успішно використовується підприємствами різних масштабів і галузей. Вона надає гнучкі інструменти для управління ключовими бізнес-процесами, допомагає оптимізувати операційну діяльність і сприяє досягненню стратегічних цілей організації.

1.2.2 Oracle Warehouse Management

Oracle Warehouse Management – це ERP-рішення, орієнтоване на автоматизацію управління складськими операціями та товарними запасами (рис. 1.4). Система забезпечує комплексні можливості для:

– контролю руху матеріальних ресурсів;

– обліку приймання та відвантаження товарів;

– прогнозування попиту й планування запасів.

Важливою перевагою OWM є її здатність до інтеграції з іншими корпоративними системами, зокрема з платформами, що використовують технології радіочастотної ідентифікації (RFID). Це дозволяє досягти більшої узгодженості між процесами постачання, зберігання та реалізації продукції.

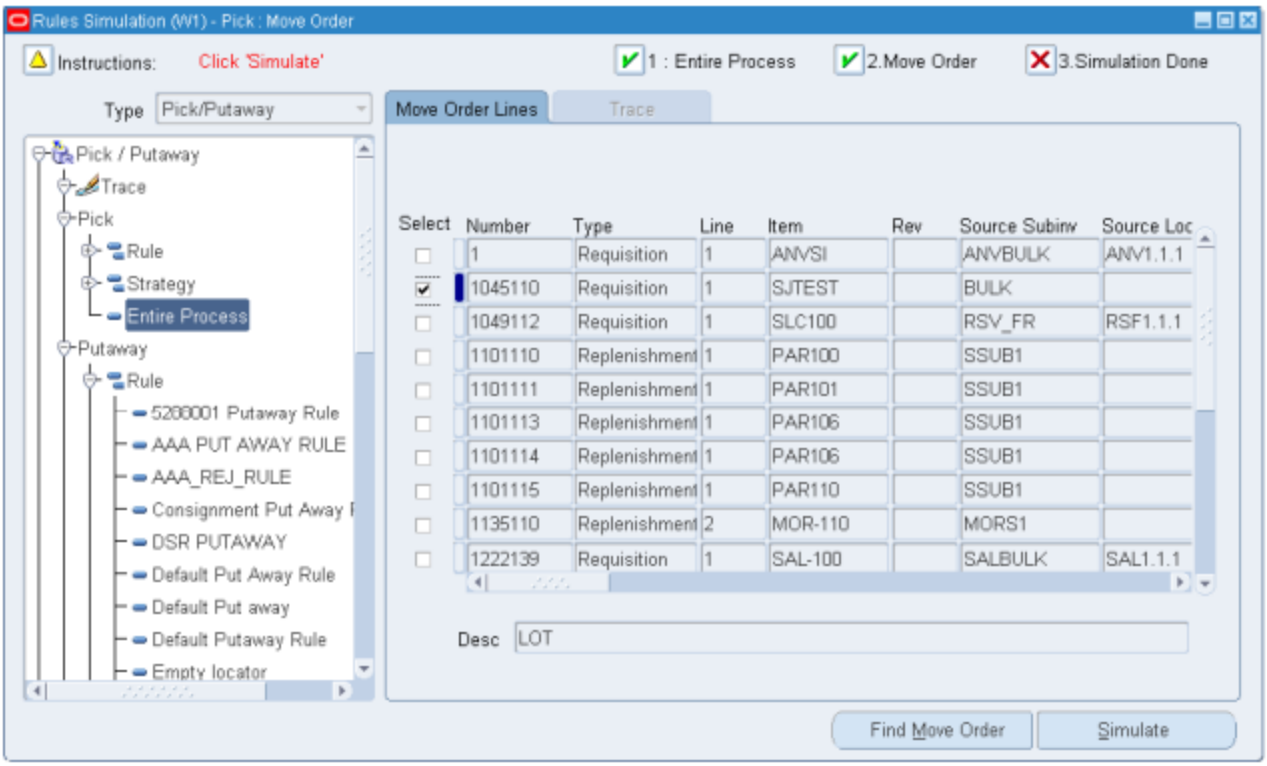


Рисунок 1.4 – Інтерфейс Oracle Warehouse Management

Попри високу ефективність, система вирізняється складністю впровадження та обслуговування. Процес інтеграції OWM потребує ретельного аналізу бізнес-процесів підприємства, налаштування функціоналу під його специфіку, поєднання з іншими інформаційними системами та підготовки персоналу до роботи з новим середовищем. Такий підхід вимагає значних часових і фінансових ресурсів, однак у підсумку забезпечує підвищення прозорості та точності складських операцій [14].

1.2.3 Microsoft Dynamics 365 Supply Chain Management

Microsoft Dynamics 365 Supply Chain Management – це ERP-рішення, орієнтоване на комплексне керування запасами, логістичними процесами та операційною діяльністю підприємства. Система надає інструменти для:

– прогнозування попиту;

– планування виробничих процесів;

– контролю логістичних потоків і управління ресурсами.

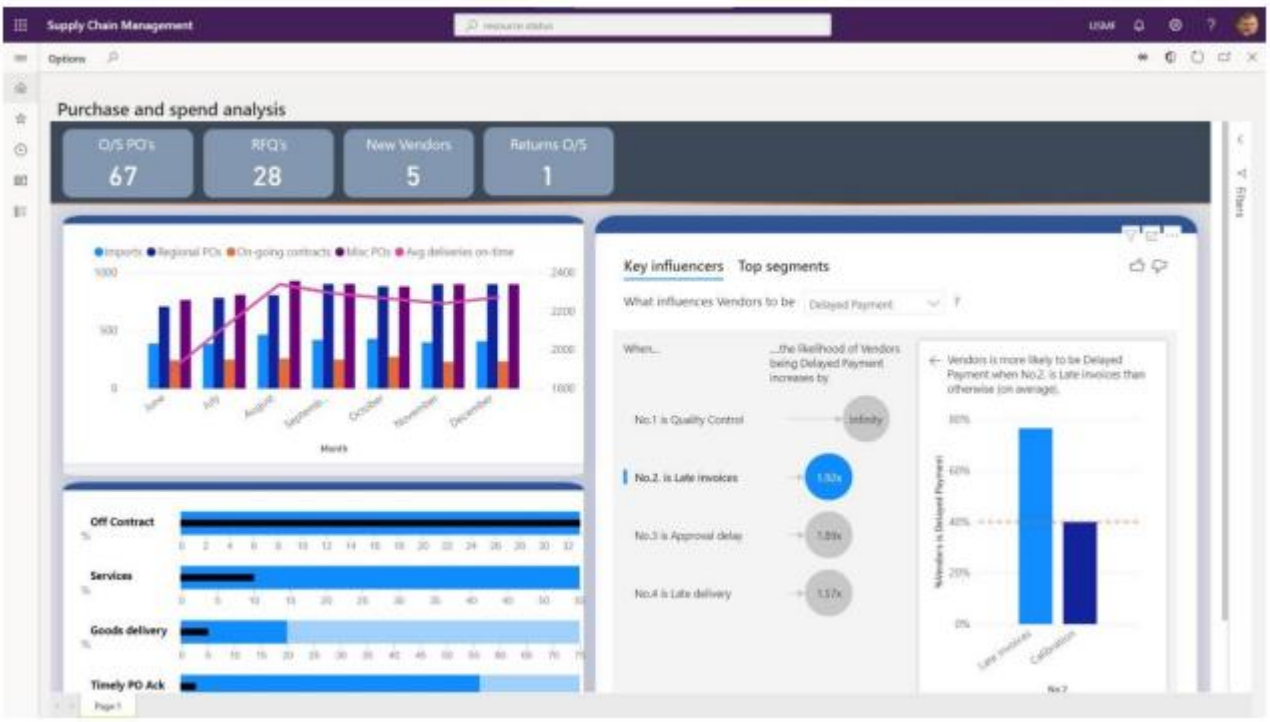


Рисунок 1.5 – Інтерфейс Microsoft Dynamics 365 Supply Chain Management

Однією з ключових переваг є тісна інтеграція з іншими продуктами екосистеми Microsoft – зокрема з Microsoft Office та Power BI, що спрощує аналітику, звітність і взаємодію між підрозділами. Завдяки цьому Dynamics 365 вважається більш зручним і доступним у використанні рішенням, порівняно з системами SAP чи Oracle, особливо для середніх і малих компаній.

Разом із тим, система має певні обмеження щодо адаптації під специфіку підприємства. Обмежений набір інструментів конфігурації та модифікації може

знижувати гнучкість під час впровадження, що ускладнює налаштування системи для організацій із нестандартними бізнес-процесами.

1.2.4 Порівняльна характеристика існуючих ERP систем

Кожна ERP-система має свої сильні сторони та обмеження, тому вибір оптимального рішення для управління складським обліком залежить від конкретних умов діяльності підприємства, його фінансових можливостей та рівня технічної підготовки. Якщо організація обмежена в бюджеті, доцільно розглянути Microsoft Dynamics 365 Supply Chain Management – систему, що відзначається нижчою вартістю впровадження та простішою експлуатацією. У випадку ж великих міжнародних корпорацій, яким необхідна підтримка кількох мов і валют, доцільним буде впровадження SAP, яка краще пристосована до глобальних бізнес-процесів [15].

Вибір платформи управління ланцюгом постачання повинен базуватись на глибокому аналізі внутрішніх потреб компанії. Під час прийняття рішення важливо оцінити такі чинники, як:

– функціональні можливості системи;

– вартість ліцензії та технічного обслуговування;

– складність процесу впровадження;

– можливості інтеграції з наявними програмними рішеннями;

– рівень технічної підтримки та масштабованості.

Окрім технічних параметрів, суттєве значення має зручність інтерфейсу та інтуїтивність роботи користувачів із системою, особливо в умовах великої кількості персоналу (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика існуючих ERP систем

Особливості	Odoo	SAP	Microsoft Dynamics 365 Supply Chain Management	Oracle Warehouse Management
Функціональність	Повний набір функцій	Широкий спектр функцій	Розширені можливості управління ланцюгом постачання	Широкий спектр функцій
Вартість	Відкритий код, вартість реалізації та підтримки залежить від постачальника та потреб підприємства	Висока вартість володіння та підтримки	Висока вартість володіння та підтримки	Висока вартість володіння та підтримки
Впровадження	Відносно простий процес впровадження	Складний та часоємний процес впровадження	Складний та часоємний процес впровадження	Складний та часоємний процес впровадження
Гнучкість	Висока гнучкість і настроюваність	Висока гнучкість та можливості настроювання	Обмежені можливості настроювання	Обмежені можливості настроювання

Таким чином, для успішного вибору ERP-рішення підприємству необхідно всебічно дослідити ринок, порівняти можливі варіанти та обрати систему, яка

найбільш точно відповідає його стратегічним цілям, ресурсам і особливостям бізнес-процесів.

1.3 Огляд комерційних WMS систем та їх порівняльна характеристика

1.3.1 Огляд ринку автоматизованих систем управління складом

Система управління складом (Warehouse Management System – WMS) – це програмний комплекс, призначений для організації, контролю та координації всіх процесів, що відбуваються під час зберігання і переміщення матеріальних ресурсів. Така система забезпечує взаємодію між технічними засобами, програмними модулями та користувачами, створюючи єдину інформаційну платформу управління складом.

До складу типової системи входять такі основні елементи:

– прикладний рівень, що виконує функції людино-машинного інтерфейсу та забезпечує оператора можливістю вводити, змінювати й аналізувати дані, формувати звіти та виконувати керуючі дії; програмне забезпечення може встановлюватися на комп’ютер, планшет або мобільний пристрій;

– серверна частина, яка відповідає за зберігання, оброблення, передавання та захист інформації, отриманої від клієнтських застосунків;

– рівень бізнес-логіки, що реалізує алгоритми оброблення запитів, формування звітів, оновлення бази даних і повідомлення користувача про результати виконаних дій.

Під час внутрішніх операцій переміщення вантажів система автоматично формує оптимальні маршрути, що дозволяють підвищити ефективність використання транспортного обладнання та мінімізувати непотрібні поїздки. Якщо на складі застосовуються різні типи навантажувальної техніки, програма обирає найдоцільніший варіант для кожного завдання. Після завершення операції система зчитує ідентифікаційну мітку вантажу, що запобігає помилкам і забезпечує своєчасне оновлення інформації у базі даних.

Автоматизовані системи управління складами поділяють на кілька основних типів:

– типові (коробкові) системи;

– адаптивні системи;

– індивідуальні (на замовлення) рішення.

Типові системи – це готові програмні продукти, які постачаються у стандартній конфігурації та передбачають обмежене налаштування під потреби користувача. Попри свою простоту, вони забезпечують необхідний рівень автоматизації базових операцій і можуть швидко впроваджуватися без суттєвих витрат.

Адаптивні системи створюються для підприємств, яким недостатньо стандартного функціоналу, але розроблення повністю індивідуального рішення є економічно невиправданим. Такі системи характеризуються гнучкістю налаштувань, підтримують різні операційні середовища й бази даних, а їхня вартість залежить від масштабів упровадження та кількості користувачів.

Індивідуальні системи орієнтовані на специфічні потреби окремого підприємства та охоплюють повний цикл логістичних процесів. Їх розроблення здійснюється на основі базових модулів, які розширюються індивідуальними функціональними компонентами. Такі системи зазвичай реалізуються протягом тривалого часу, потребують значних інвестицій, проте забезпечують максимальну відповідність структурі та технологічним особливостям організації.

1.3.2 Порівняльна характеристика існуючих WMS-систем

На рисунку 1.6 наведено схематичний розподіл компаній на світовому ринку систем управління складом. Для правильного тлумачення наведеної діаграми слід враховувати такі орієнтири класифікації:

– «лідери» – постачальники, які мають високі оцінки за обома критеріями: ефективністю реалізації рішень і повнотою бачення розвитку технологій;

– «кандидати» – компанії, що демонструють сильні позиції лише за показником здатності реалізовувати рішення на практиці;

– «провидці» – розробники, які мають позитивну оцінку за стратегічним баченням, але ще не досягли високої результативності у впровадженні [16];

– «нішеві гравці» – постачальники з обмеженими можливостями у реалізації продуктів та недостатньо розвиненим стратегічним баченням ринку .

Рисунок 1.6 – Квадрант розподілу компаній на ринку складських систем

В таблиці 1.2 наведена порівняльна характеристика провідних WMS-систем.

Таблиця 1.2 – Порівняльна характеристика провідних WMS-систем

Назва компанії	Надані можливості системи
Blue Yonder (раніше JDA)	– забезпечує високу прозорість і точне щоденне планування в інтегрованій мережі постачань і продажів; – підтримує різні робочі процеси та виробничі середовища; – можливість формування звітів за винятковими ситуаціями; – реалізує моделювання сценаріїв типу «що, якщо»; – містить галузеві моделі для вибору найбільш економічного рішення; – підтримує бізнес-правила для капіталомістких виробництв.
Infor	– забезпечує повний і надійний функціонал для всіх процесів промислового підприємства; – дозволяє швидко й точно планувати завантаження обладнання, трудові витрати та потреби у сировині; – підтримує контроль виробництва відповідно до робочих завдань, планів або етикеток KANBAN; – забезпечує швидке опрацювання клієнтських замовлень і формує реалістичні виробничі пропозиції.
Körber	– дозволяє вносити зміни до рішення у будь-який момент; – повністю підтримує конфігурацію системи замовника, включно з власними розробками; – зберігає всі напрацьовані дані та налаштування під час оновлення або переходу на нову версію; – містить сертифікований вбудований модуль Advantage Link, який інтегрує всі застосунки Körber із провідними світовими бізнес-системами.
Manhattan Associates	– забезпечує оцінку ефективності персоналу; – підтримує сучасні технології управління складом: голосове керування, RFID, інтеграцію обладнання; – оптимізує розміщення товарів на складі; – дозволяє керувати завданнями працівників і контролювати навантаження персоналу; – підтримує контроль усіх складських операцій; – спрощує документообіг і підвищує ефективність взаємодії з контрагентами; – забезпечує виконання всіх операцій за допомогою радіотерміналів.

1.4 Висновки до розділу

У першому розділі виконано аналіз сучасного стану засобів автоматизації складської логістики та обґрунтовано актуальність розробки власної WMS-підсистеми для логістичного центру підприємства. Дослідження показало, що впровадження автоматизованих систем управління складом є необхідною умовою ефективної роботи підприємств в умовах зростання обсягів виробництва і товарообігу.

Проаналізовано концепції ERP та WMS: встановлено, що WMS-системи еволюціонували від простих інструментів обліку запасів до комплексних інтелектуальних платформ, здатних у реальному часі відстежувати запаси, оптимізувати процеси розміщення товарів, планувати ресурси та підтримувати ухвалення управлінських рішень на основі даних. Особливу увагу приділено сучасним тенденціям – інтеграції технологій Інтернету речей (IoT) на складах (використання RFID для ідентифікації товарів, сенсорів для моніторингу умов зберігання) та застосуванню алгоритмів штучного інтелекту для прогнозування потреб складу і підвищення адаптивності системи. У рамках огляду існуючих рішень було розглянуто можливості провідних комерційних WMS-продуктів (SAP, Oracle, Microsoft Dynamics та ін.) і виявлено їх спільні риси: широкий функціонал, висока надійність, але водночас значна вартість впровадження та потенційні труднощі адаптації під специфічні процеси підприємства. Альтернативою є використання платформ з відкритим кодом, зокрема ERP-платформи Odoo, яка виділяється своєю гнучкістю та модульністю. За результатами порівняльного аналізу встановлено, що Odoo як WMS-платформа має низку переваг: можливість кастомізації без зміни ядра системи, інтеграція складського модуля з іншими підсистемами (закупівлі, продажі, виробництво, фінанси) в єдиному інформаційному просторі, відсутність ліцензійних витрат. Відзначено, що використання Odoo потребує менших прямих інвестицій, проте може вимагати додаткових зусиль на розробку специфічного функціоналу та підтримку.

2 ВИБІР ТЕХНОЛОГІЙ ТА СЕРЕДОВИЩА РОЗРОБКИ

2.1 Обґрунтування вибору Odoo як WMS-платформи

1.2.1 Платформа Odoo

Odoo є комплексною програмною платформою з відкритим вихідним кодом, призначеною для автоматизації та інтегрованого управління бізнес-процесами підприємства. Архітектура системи побудована за модульним принципом, що дозволяє охопити ключові функціональні напрями діяльності компанії, зокрема складську логістику, управління продажами та закупівлями, фінансовий облік, виробниче планування, управління проєктами та інші процеси.

Важливою перевагою Odoo є відкритість програмного коду, що надає можливість глибокої адаптації системи під специфічні вимоги користувача. Це дозволяє змінювати логіку роботи модулів, розширювати функціональність та інтегрувати власні рішення без обмежень, характерних для комерційних закритих систем. Додатково система характеризується інтуїтивно зрозумілим інтерфейсом (рис. 2.1), що знижує поріг входження для персоналу та мінімізує потребу у тривалому навчанні.

Суттєвим фактором при виборі Odoo є наявність розвиненої міжнародної спільноти розробників і користувачів, яка забезпечує постійний розвиток платформи, оперативне усунення помилок та регулярне оновлення модулів відповідно до сучасних вимог цифрової трансформації бізнесу.

Завдяки високому рівню гнучкості Odoo може бути конфігурований для вирішення широкого спектра управлінських задач, у тому числі організації складського обліку та логістичних операцій у форматі WMS. Водночас слід зазначити, що процес впровадження та тонкого налаштування системи може

потребувати залучення фахівців з відповідними знаннями у сфері програмування та адміністрування інформаційних систем [17].

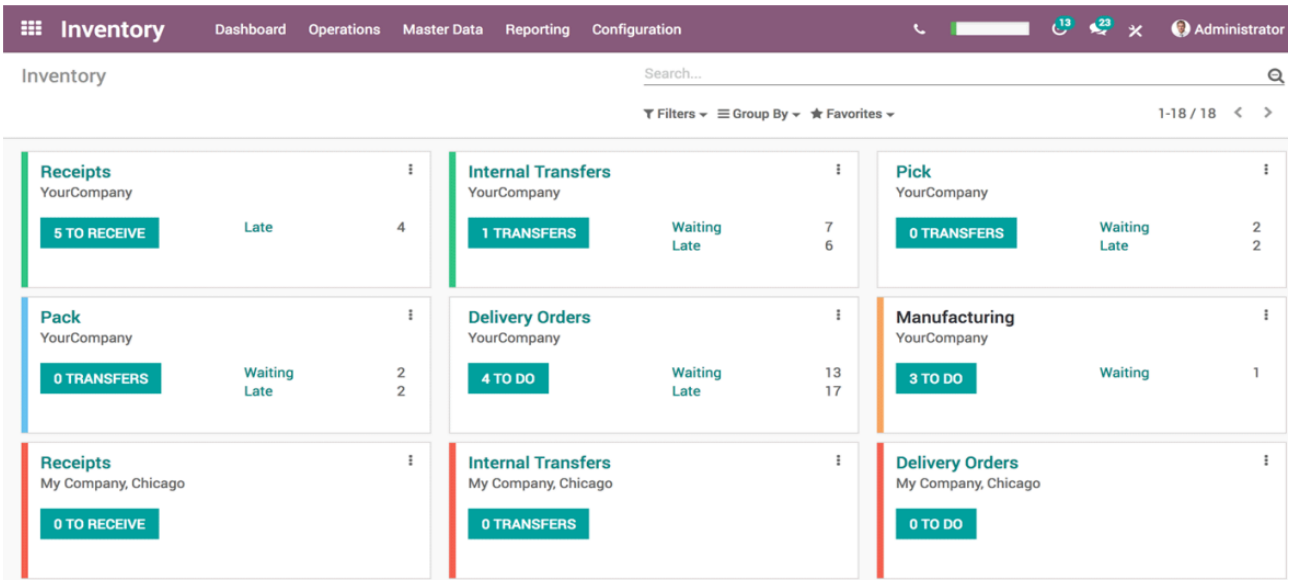


Рисунок 2.1 – Приклад інтерфейсу Odoo

Платформа підтримує багатомовність та механізми локалізації, що забезпечує її застосування в міжнародному середовищі з урахуванням національних нормативно-правових та культурних особливостей. Додатково Odoo включає інструменти для створення та керування електронною комерцією, а також підтримує інтеграцію з зовнішніми сервісами – платіжними шлюзами, службами доставки, поштовими сервісами та іншими інформаційними системами.

Разом з тим, Odoo не позбавлена обмежень. За відсутності належного досвіду адміністрування можливі складнощі з налаштуванням безпеки та оптимізацією продуктивності. При значному масштабуванні системи та зростанні кількості одночасних користувачів можуть виникати вимоги до додаткових обчислювальних ресурсів і оптимізації серверної інфраструктури.

Узагальнюючи, Odoo можна охарактеризувати як функціонально насичену та адаптивну платформу управління бізнесом, яка за умови коректного впровадження здатна ефективно виконувати роль WMS-системи та

забезпечувати інтеграцію складського обліку з іншими підсистемами підприємства.

2.2 Реалізація WMS-функціональності на основі платформи Odoo

Платформа Odoo реалізує концепцію інтегрованого управління підприємством, у межах якої складська логістика функціонує у тісному взаємозв’язку з підсистемами продажів, постачання, фінансового обліку та виробничого планування. Такий підхід забезпечує узгодженість дій між структурними підрозділами організації та сприяє зменшенню інформаційних розривів між бізнес-процесами, що позитивно впливає на загальну продуктивність та ефективність діяльності підприємства.

Функціональні можливості Odoo у сфері складської логістики охоплюють контроль складських залишків, операції приймання та відвантаження продукції, формування і обробку замовлень, процеси пакування, а також моніторинг руху товарів на етапах доставки. Використання вбудованих механізмів автоматизації дозволяє скоротити кількість ручних операцій, мінімізувати ймовірність помилок персоналу та підвищити оперативність виконання складських процедур.

Важливою характеристикою Odoo є її гнучкість та можливість адаптації до умов конкретного підприємства. Система підтримує застосування стандартних модулів складського обліку, однак їх базовий функціонал не завжди охоплює повний перелік вимог, що висуваються до сучасних спеціалізованих WMS-систем. У таких випадках виникає необхідність у розширенні або доопрацюванні наявних модулів відповідно до специфіки логістичних процесів підприємства.

Як універсальна ERP-платформа, Odoo може поступатися спеціалізованим WMS-рішенням за рівнем галузевої спрямованості та глибиною реалізації окремих складських функцій. Це може становити обмеження для підприємств зі складною логістичною інфраструктурою або з нестандартними вимогами до управління запасами та товаропотоками.

Разом з тим, відкритий програмний код Odoo надає можливість розроблення та інтеграції власних модулів, що дозволяє реалізувати індивідуальні алгоритми управління складськими процесами. Такий підхід забезпечує повний контроль підприємства над інформаційною системою, зменшує залежність від ліцензійних обмежень та створює умови для подальшого функціонального розвитку системи відповідно до потреб бізнесу.

До недоліків використання Odoo як WMS-рішення слід віднести потенційно підвищені витрати на впровадження, налаштування, технічну підтримку та навчання персоналу. У порівнянні з окремими готовими спеціалізованими WMS-системами, сумарна вартість володіння може бути вищою, особливо за умови масштабування системи або необхідності значних доопрацювань.

Узагальнюючи, Odoo може розглядатися як ефективна платформа для побудови системи управління складом за умови її цілеспрямованої адаптації. Вона забезпечує гнучкі інструменти автоматизації та створює передумови для оптимізації складських операцій у межах єдиного інформаційного простору підприємства [18].

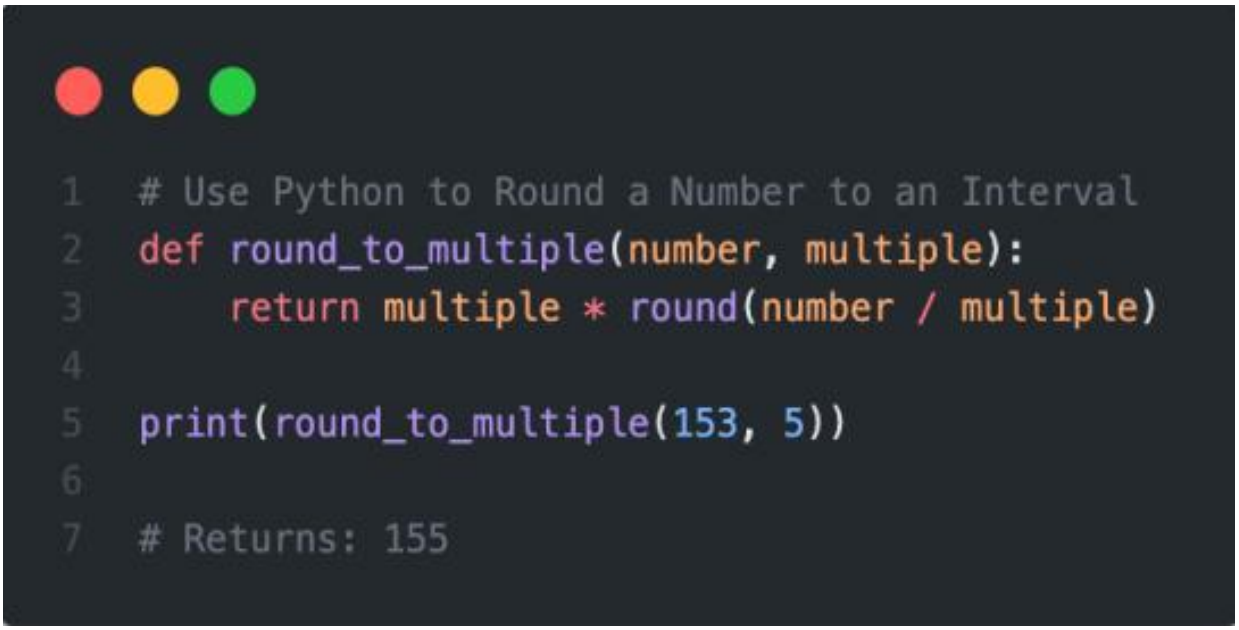
2.3 Обрані технології розробки

2.3.1 Мова програмування Python

Python є високорівневою інтерпретованою мовою програмування загального призначення, яка поєднує лаконічність синтаксису з потужним функціональним інструментарієм. Мова була створена Гвідо ван Россумом і вперше представлена у 1991 році, після чого набула широкого поширення в наукових, інженерних та прикладних сферах.

Однією з ключових переваг Python є його простота та низький поріг входження. Чітка та логічна структура синтаксису забезпечує високу читабельність програмного коду (рис. 2.2), що суттєво полегшує процес розробки, супроводу та подальшої модифікації програмних рішень. Завдяки

цьому Python активно використовується як початківцями, так і досвідченими розробниками.



```python
# Use Python to Round a Number to an Interval
def round_to_multiple(number, multiple):
    return multiple * round(number / multiple)

print(round_to_multiple(153, 5))

# Returns: 155
```

Рисунок 2.2 – Приклад коду на мові Python

Python належить до мов загального призначення, що дозволяє застосовувати його для вирішення широкого кола задач. Зокрема, мова активно використовується у веб-розробці, аналізі та обробці даних, наукових дослідженнях, системах штучного інтелекту, машинному навчанні, автоматизації бізнес-процесів та розробці серверних застосунків.

Важливою особливістю Python є наявність розвиненої екосистеми стандартних бібліотек і сторонніх модулів, які значно розширюють базовий функціонал мови. Активна міжнародна спільнота розробників постійно підтримує та оновлює ці бібліотеки, що дозволяє швидко реалізовувати складні програмні рішення без необхідності створення всіх компонентів з нуля.

Мова Python використовує механізм динамічної типізації, який передбачає визначення типів змінних під час виконання програми. Це підвищує гнучкість програмного коду та прискорює процес розробки, особливо на етапах прототипування та тестування.

Крім того, Python підтримує декілька парадигм програмування, зокрема процедурну, об’єктно-орієнтовану та функціональну. Така універсальність дозволяє адаптувати стиль програмування до специфіки задачі та архітектури програмної системи.

Узагальнюючи, Python є ефективною, гнучкою та масштабованою мовою програмування, яка забезпечує широкі можливості для створення сучасних програмних продуктів. Вибір Python у даній роботі також обумовлений тим, що платформа Odoo реалізована саме з використанням цієї мови, що забезпечує природну інтеграцію та спрощує розробку і модифікацію WMS-функціональності.

2.3.2 Фреймворк OWL

У межах платформи Odoo абревіатура OWL означає Object Widget Library та використовується для позначення сучасного клієнтського фреймворку, призначеного для побудови користувацького інтерфейсу. OWL являє собою набір компонентів і віджетів, що застосовуються для формування динамічних елементів інтерфейсу у веб-застосунках Odoo.

Фреймворк OWL надає розробникам інструменти для створення широкого спектра інтерфейсних компонентів, зокрема кнопок, полів введення, списків, таблиць, фільтрів та інших інтерактивних елементів. Завдяки компонентному підходу забезпечується зручність повторного використання коду (рис. 2.3), а також підвищується гнучкість і масштабованість клієнтської частини системи.

Важливою особливістю OWL є його тісна інтеграція з серверною частиною Odoo. Для обміну даними між клієнтом і сервером використовується асинхронна взаємодія на основі технологій AJAX, що дозволяє оновлювати вміст сторінок без їх повного перезавантаження. Це забезпечує високу швидкодію інтерфейсу та оперативну реакцію системи на дії користувача в реальному часі.

```
odoo.define("slide_local_video.form_render", function (require) {
    "use strict";


    var FormRenderer = require('web.FormRenderer');
    var FormController = require('web.FormController');
    var FormView = require('web.FormView');
    FormRenderer.include({
        _renderView: function () {
        var self = this;
        var res = this._super.apply(this, arguments).then(function () {
            $(self.$('#input_file')).on('change', function (el) {
                self.readFile(el.currentTarget.files[0], self)
            })
        });

            return res
    },
```

Рисунок 2.3 – Приклад коду на мові JS з використанням OWL

OWL підтримує механізми розширення та модифікації поведінки компонентів за допомогою класів і сучасних можливостей мови JavaScript. Такий підхід дозволяє реалізовувати складну бізнес-логіку на стороні клієнта, адаптувати стандартні віджети під специфічні вимоги та створювати власні інтерактивні елементи інтерфейсу.

Узагальнюючи, фреймворк OWL відіграє ключову роль у формуванні сучасного, адаптивного та зручного користувацького інтерфейсу в системі Odoo. Його використання сприяє підвищенню ергономіки роботи з системою, покращенню користувацького досвіду та ефективності взаємодії з бізнес-функціональністю платформи.

2.3.3 Мова розмітки XML

XML (Extensible Markup Language) є універсальною розширюваною мовою розмітки, призначеною для формалізованого подання структурованих даних у цифровому вигляді. Вона широко застосовується для зберігання, обміну та інтеграції інформації між різними програмними системами, а також у веб-

застосунках, сервіс-орієнтованих архітектурах і корпоративних інформаційних системах (рис. 2.4).

Ключовою характеристикою XML є можливість самостійного визначення структури даних. Розробник може створювати власні теги та ієрархічні зв’язки між ними, що забезпечує гнучкість у моделюванні інформації відповідно до конкретних вимог прикладної задачі. Такий підхід робить XML придатним для опису складних об’єктів і взаємопов’язаних даних.

Завдяки текстовому формату XML-документи є зручними для читання, аналізу та редагування як людиною, так і програмними засобами. Це спрощує процес налагодження, супроводу та інтеграції систем. Важливою особливістю є також розділення логічної структури даних та їх візуального подання: інформація зберігається у вигляді структурованих елементів, тоді як відображення може визначатися окремо за допомогою стилів або трансформацій.

XML є платформонезалежним стандартом, що дозволяє використовувати його для обміну даними між різними операційними системами, мовами програмування та програмними середовищами. Саме ця властивість зробила XML одним з базових форматів у міжсистемній взаємодії та корпоративних інформаційних рішеннях.

У контексті фреймворку Odoo мова XML відіграє ключову роль у конфігурації та розширенні функціональності системи. За допомогою XML-файлів описуються структури моделей даних, форми користувацького інтерфейсу, представлення списків і звітів, правила доступу, меню, дії та елементи бізнес-логіки. Такий підхід дозволяє відокремити прикладну логіку від опису інтерфейсу та конфігураційних параметрів, що спрощує підтримку і масштабування модулів.

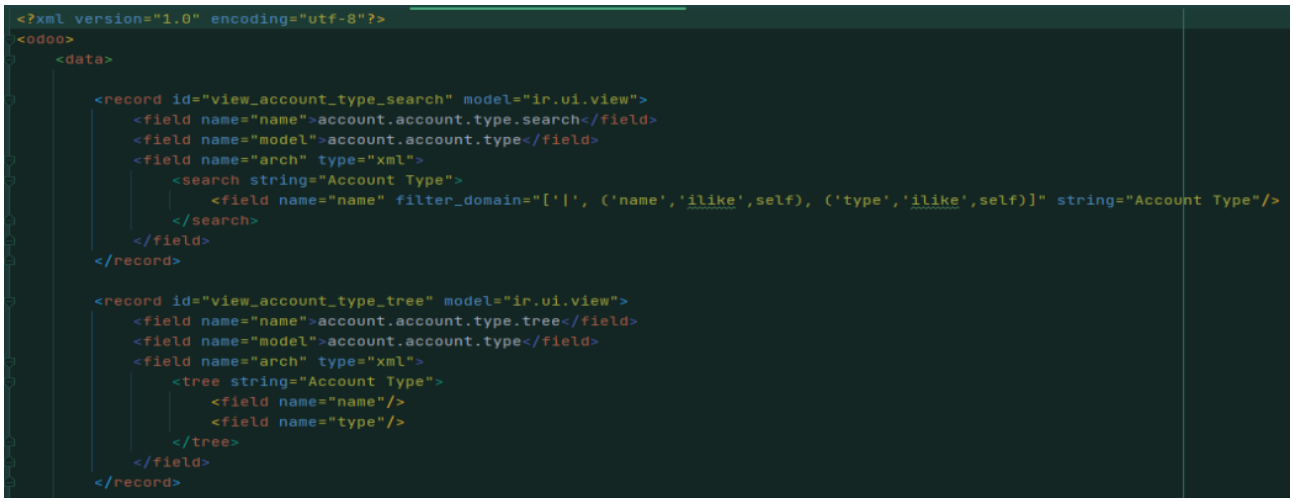


Рисунок 2.4 – Приклад коду мовою XML

Узагальнюючи, XML є ефективним засобом структуризації та формального опису даних. У системі Odoo використання XML забезпечує гнучке налаштування модулів, уніфікований опис інтерфейсу та логіки доступу, що суттєво спрощує процес розробки, адаптації та супроводу програмних компонентів.

2.3.4 Фреймворк Vue.js

Vue.js є сучасним прогресивним JavaScript-фреймворком, призначеним для розробки користувацьких інтерфейсів веб-застосунків. Основна концепція Vue.js полягає у створенні реактивних односторінкових додатків (Single Page Applications, SPA), у яких інтерфейс динамічно оновлюється без перезавантаження сторінки, що забезпечує підвищену зручність та швидкодію взаємодії з користувачем.

Однією з важливих переваг Vue.js є відносна простота його освоєння. Фреймворк використовує зрозумілий та логічний синтаксис, що дозволяє швидко перейти від базових прикладів до створення повноцінних веб-інтерфейсів. Наявність детальної офіційної документації та активної спільноти розробників сприяє оперативному вирішенню технічних питань і впровадженню кращих практик розробки.

Vue.js базується на реактивній моделі програмування, у межах якої будь-які зміни стану даних автоматично відображаються в інтерфейсі користувача. Такий підхід значно спрощує керування станом застосунку та зменшує складність синхронізації між логікою обробки даних і візуальним поданням інформації.

Архітектура Vue.js побудована на компонентному принципі, що передбачає розділення інтерфейсу на незалежні функціональні блоки. Компоненти можуть бути багаторазово використані в різних частинах застосунку, що спрощує супровід коду, підвищує його модульність та знижує витрати часу на розробку і тестування.

Фреймворк має розвинену екосистему інструментів, плагінів і доповнень, які дозволяють розширювати базовий функціонал та інтегрувати Vue.js з іншими бібліотеками й серверними технологіями. Це забезпечує гнучкість у виборі архітектурних рішень та дозволяє застосовувати Vue.js як у невеликих проєктах, так і у складних корпоративних вебсистемах.

Завдяки оптимізованим механізмам рендерингу та підтримці відкладеного завантаження компонентів Vue.js демонструє високу продуктивність навіть при роботі з великими обсягами даних. Це позитивно впливає на швидкість відгуку інтерфейсу та загальний користувацький досвід.

Узагальнюючи, Vue.js є ефективним та універсальним JavaScript-фреймворком для створення реактивних веб-інтерфейсів. Поєднання простоти використання, реактивної моделі та компонентної архітектури робить його популярним інструментом для розробки сучасних веб-застосунків і клієнтських частин інформаційних систем.

2.3.5 Система управління базами даних PostgreSQL

PostgreSQL є сучасною об’єктно-реляційною системою управління базами даних, що забезпечує надійне зберігання, обробку та керування великими обсягами структурованої інформації. Вона належить до високопродуктивних СУБД з відкритим вихідним кодом та широко використовується у

корпоративних інформаційних системах, веб-застосунках і системах автоматизації бізнес-процесів.

Архітектура PostgreSQL базується на реляційній моделі даних, у межах якої інформація організовується у вигляді таблиць із чітко визначеними зв’язками між ними. Такий підхід забезпечує логічну впорядкованість даних, спрощує їх аналіз та дозволяє ефективно реалізовувати складні запити до бази даних.

Система повністю підтримує стандарт мови SQL та розширює його власними можливостями. PostgreSQL надає розвинений набір інструментів для формування запитів, об’єднання таблиць, агрегування даних, використання вкладених підзапитів і збережених процедур. Це дозволяє реалізовувати складну бізнес-логіку безпосередньо на рівні бази даних.

До важливих переваг PostgreSQL належать механізми підтримки транзакцій, забезпечення цілісності даних, оптимізації виконання запитів та індексації. Система підтримує різноманітні типи індексів, що підвищує швидкодію при роботі з великими масивами інформації. Також реалізовані засоби резервного копіювання, реплікації та відновлення даних, що підвищує загальний рівень надійності.

PostgreSQL характеризується високою розширюваністю. Вона дозволяє створювати власні функції, типи даних, оператори та навіть підключати додаткові мови програмування для обробки даних. Це забезпечує можливість адаптації СУБД до специфічних вимог прикладних систем і реалізації нестандартних рішень.

Завдяки підтримці масштабування та ефективній роботі з великими обсягами даних PostgreSQL може застосовуватися як у невеликих проєктах, так і у високонавантажених корпоративних системах. Реалізовані механізми контролю доступу та захисту інформації дозволяють використовувати систему в середовищах з підвищеними вимогами до безпеки.

У контексті розробки та експлуатації платформи Odoo PostgreSQL використовується як основна система управління базами даних для зберігання

прикладної, конфігураційної та транзакційної інформації. Поєднання надійності, продуктивності та гнучкості робить PostgreSQL доцільним вибором для автоматизації бізнес-процесів і реалізації WMS-функціональності на базі Odoo.

2.4 Ознайомлення з середовищем для розробки програмного додатку

2.4.1 Інтегроване середовище розробки PyCharm

PyCharm є інтегрованим середовищем розробки для мови програмування Python, створеним компанією JetBrains, яке орієнтоване на підвищення ефективності та зручності процесу програмування (рис. 2.5). Дане IDE поєднує в собі інструменти для написання, аналізу, налагодження та супроводу програмного коду, що робить його доцільним вибором для розробки прикладних і корпоративних Python-застосунків.

Рисунок 2.5 – PyCharm

Однією з ключових складових PyCharm є розвинений редактор коду, який підтримує підсвічування синтаксису, інтелектуальне автодоповнення, статичний аналіз коду та автоматичне виявлення помилок. Редактор забезпечує роботу не

лише з Python-файлами, але й з іншими форматами, що використовуються у веб-розробці та серверних застосунках, зокрема HTML, CSS і JavaScript.

Середовище PyCharm надає зручні засоби керування проєктами, що дозволяє структурувати програмний код, керувати залежностями, створювати та налаштовувати віртуальні середовища, а також конфігурувати параметри запуску застосунків. Це спрощує організацію великих програмних проєктів і сприяє підтримці впорядкованої архітектури коду.

Важливу роль у PyCharm відіграє інструментарій налагодження програм. Середовище підтримує встановлення точок зупинки, покрокове виконання коду, аналіз значень змінних у режимі реального часу та дослідження стеку викликів. Такі можливості значно полегшують процес виявлення та усунення логічних і синтаксичних помилок.

PyCharm також забезпечує інтелектуальні механізми автозавершення коду, які аналізують контекст програми та пропонують доступні класи, методи, функції й атрибути. Це дозволяє зменшити кількість помилок при написанні коду та підвищити швидкість розробки.

Додатковою перевагою PyCharm є інтеграція з зовнішніми інструментами розробки. Середовище підтримує роботу з системами контролю версій, зокрема Git, менеджерами пакетів Python, засобами тестування та базами даних. Завдяки цьому розробник може виконувати більшість необхідних операцій безпосередньо в межах одного інтерфейсу.

Функціональність PyCharm може бути розширена за допомогою плагінів, які дозволяють додавати підтримку додаткових мов програмування, інструментів аналізу коду та сторонніх сервісів. Це забезпечує гнучкість середовища та можливість адаптації під специфічні вимоги конкретного проєкту.

Узагальнюючи, PyCharm є потужним і багатофункціональним середовищем розробки, яке забезпечує високий рівень продуктивності при створенні Python-додатків. Його використання сприяє підвищенню якості програмного коду, скороченню часу розробки та зручному супроводу програмних систем.

2.4.2 Середовище адміністрування баз даних pgAdmin 4

pgAdmin 4 є спеціалізованим графічним програмним засобом, призначеним для адміністрування та керування системами управління базами даних PostgreSQL (рис. 2.6). Дане середовище забезпечує користувачу зручний інтерфейс для виконання повного спектра операцій, пов’язаних зі створенням, налаштуванням і супроводом баз даних, без необхідності постійної роботи через командний рядок.

Рисунок 2.6 – Інтерфейс PgAdmin 4

Однією з основних функціональних можливостей pgAdmin 4 є централізоване керування об’єктами бази даних. За допомогою цього інструмента здійснюється створення, редагування та видалення баз даних, схем, таблиць, індексів, представлень і тригерів. Графічний інтерфейс дозволяє виконувати ці операції у структурованому вигляді, що спрощує адміністрування складних інформаційних систем.

pgAdmin 4 також надає вбудований редактор для виконання SQL-запитів, який дозволяє напряму взаємодіяти з базою даних. Користувач має можливість формувати та виконувати запити різного типу, зокрема запити вибірки, вставки, оновлення та видалення даних, а також застосовувати розширені можливості PostgreSQL, включаючи функції, агрегатні операції та вкладені запити.

Важливою перевагою середовища є наочна візуалізація структури бази даних. pgAdmin 4 реалізує ієрархічне деревовидне подання, яке відображає взаємозв'язки між схемами, таблицями, стовпцями та іншими об'єктами. Це дозволяє швидко орієнтуватися у структурі даних, аналізувати їх організацію та отримувати детальну інформацію про властивості кожного елемента.

Окремий функціональний блок pgAdmin 4 призначений для аналізу продуктивності та оптимізації роботи бази даних. Засоби моніторингу дозволяють переглядати статистику виконання запитів, досліджувати плани виконання, виявляти потенційно неефективні операції та приймати рішення щодо підвищення швидкодії системи зберігання даних.

Значна увага в pgAdmin 4 приділяється питанням безпеки. Інструмент підтримує керування ролями та користувачами, налаштування прав доступу до об'єктів бази даних, а також виконання процедур резервного копіювання та відновлення інформації. Це забезпечує контроль доступу до даних і підвищує надійність експлуатації інформаційної системи.

Узагальнюючи, pgAdmin 4 є універсальним та ефективним засобом для адміністрування баз даних PostgreSQL, який поєднує широкі функціональні можливості з наочним графічним інтерфейсом. Його використання дозволяє спростити процес керування базами даних, підвищити продуктивність роботи та забезпечити стабільність функціонування програмних рішень.

2.5 Опис архітектури програмного забезпечення

Програмна платформа Odoo побудована на основі класичної трирівневої архітектурної моделі (рис. 2.7), яка широко застосовується під час створення

масштабованих та модульних бізнес-орієнтованих інформаційних систем. Такий підхід передбачає логічний поділ програмного забезпечення на окремі рівні відповідальності, що підвищує гнучкість, керованість та зручність супроводження системи.

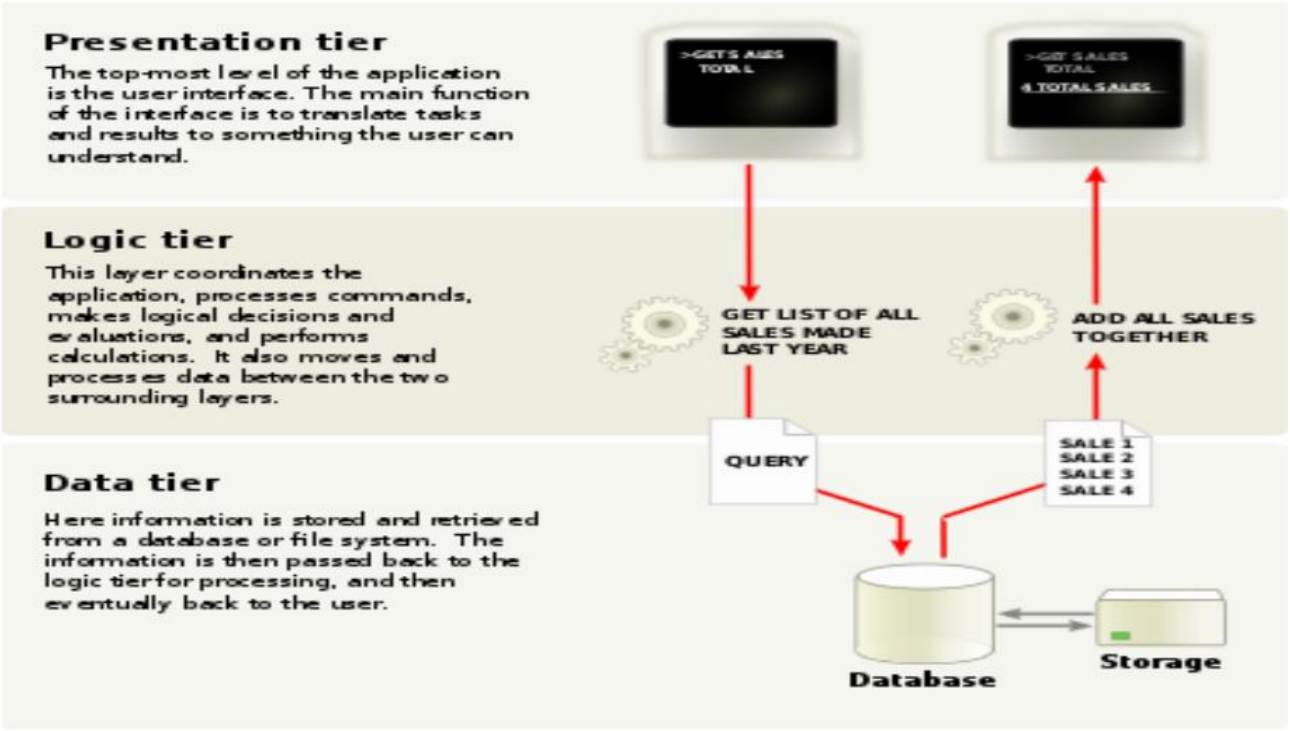


Рисунок 2.7 – Трирівнева архітектура в Odoo

Перший рівень – рівень зберігання даних.

На цьому рівні функціонує система управління базами даних PostgreSQL, яка забезпечує централізоване зберігання всієї інформації, що використовується програмною платформою. У базі даних описується структура об’єктів предметної області та зв’язки між ними, зокрема дані про контрагентів, номенклатуру, складські залишки, документи руху товарів, фінансові операції та інші сутності. Реляційна модель зберігання забезпечує цілісність, узгодженість і надійність даних.

Другий рівень – рівень прикладної (бізнес) логіки.

Даний рівень відповідає за реалізацію правил і алгоритмів функціонування бізнес-процесів підприємства. Саме тут виконуються операції обробки

замовлень, керування складськими запасами, обліку фінансових транзакцій, планування ресурсів та інших функцій. Бізнес-логіка реалізується у вигляді модулів Odoo, які можуть використовувати стандартний функціонал платформи або бути доповнені власними програмними розширеннями. Це дозволяє адаптувати систему до специфічних вимог конкретної організації без зміни базової архітектури.

Третій рівень – рівень представлення (користувацький інтерфейс).

Взаємодія кінцевих користувачів із системою здійснюється через веб-інтерфейс, доступний за допомогою браузера. Цей рівень відповідає за відображення інформації, введення даних, навігацію між функціональними модулями та забезпечення зручності роботи з системою. Для формування динамічного інтерфейсу використовуються сучасні веб-технології, зокрема JavaScript-бібліотеки та клієнтські фреймворки, що дозволяють реалізувати реактивну поведінку інтерфейсу та швидкий відгук системи.

Використання трирівневої архітектури забезпечує чітке розмежування відповідальності між зберіганням даних, обробкою бізнес-процесів і візуальним представленням інформації. Такий підхід спрощує масштабування програмного забезпечення, полегшує супровід і модернізацію системи, а також дозволяє паралельну роботу розробників над різними компонентами без порушення цілісності програмного продукту.

2.6 Структурна схема системи

На структурній схемі системи (рис. 2.8) відображено трирівневу архітектуру розробленої системи.

Перший рівень – рівень даних: це база даних (PostgreSQL), яка містить всі необхідні дані про товари, складські запаси, замовлення та операції.

Другий рівень – рівень бізнес-логіки: сервер ERP-платформи Odoo, на якому виконуються всі операції обробки даних. На цьому рівні функціонують стандартні модулі Odoo (наприклад, модуль Inventory для складського обліку) та

розроблений користувацький модуль WMS, який розширює стандартний функціонал. Логіка реалізована у вигляді набору моделей і методів Odoo, що забезпечують виконання запитів, оновлення бази даних, бізнес-правила складських процесів тощо.

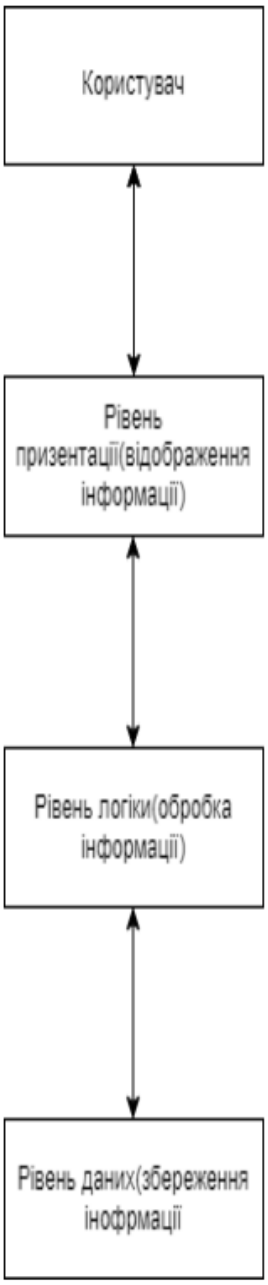


Рисунок 2.8 – Структурна схема системи

Третій рівень – рівень представлення (користувацький інтерфейс): веб-інтерфейс Odoo, через який кінцеві користувачі (оператори складу, менеджери логістики) взаємодіють із системою за допомогою браузера. Інтерфейс відповідає за відображення актуальної інформації, введення користувачем даних та навігацію між функціональними модулями.

Окрім того, до системи підключені термінали збору даних (ТЗД) – переносні пристрої (сканери), що по мережі взаємодіють із сервером Odoo. Вони дозволяють здійснювати оперативне зчитування інформації з товарних етикеток (штрих-кодів, RFID) та автоматично передавати ці дані на сервер, де вони відразу обробляються WMS-підсистемою. Таким чином, структурна схема включає в себе: центральний сервер Odoo (з WMS-модулем) як ядро системи, підключену до нього базу даних, користувачів з веб-браузерами та ТЗД у ролі клієнтів системи.

Така трирівнева структура чітко розмежовує зони відповідальності між зберіганням даних, обробкою бізнес-процесів і відображенням інформації, що спрощує масштабування і підтримку системи, а також дозволяє незалежно розвивати кожен з компонентів без негативного впливу на інші.

2.7 Функціональна схема (діаграма класів)

Функціональна схема представлена у вигляді діаграми класів, яка моделює основні інформаційні сутності системи та зв’язки між ними (рис. 2.9). У центрі моделі знаходиться клас StockPicking, що відповідає за складський документ (операцію верхнього рівня) – наприклад, відвантаження замовлення клієнту або надходження товару від постачальника. Об’єкт StockPicking агрегує набір конкретних переміщень товарів і виступає логічним об’єднанням для групи пов’язаних операцій. Безпосередньо переміщення товарів моделюються класом StockMove – кожен об’єкт цього класу описує одиничну операцію переміщення матеріальних цінностей на складі. Модель StockMove містить ключові атрибути, необхідні для контролю руху запасів: посилання на товар (Product), що

переміщується; початкову та кінцеву локації (Location) зберігання; заплановану кількість; статус виконання операції тощо.

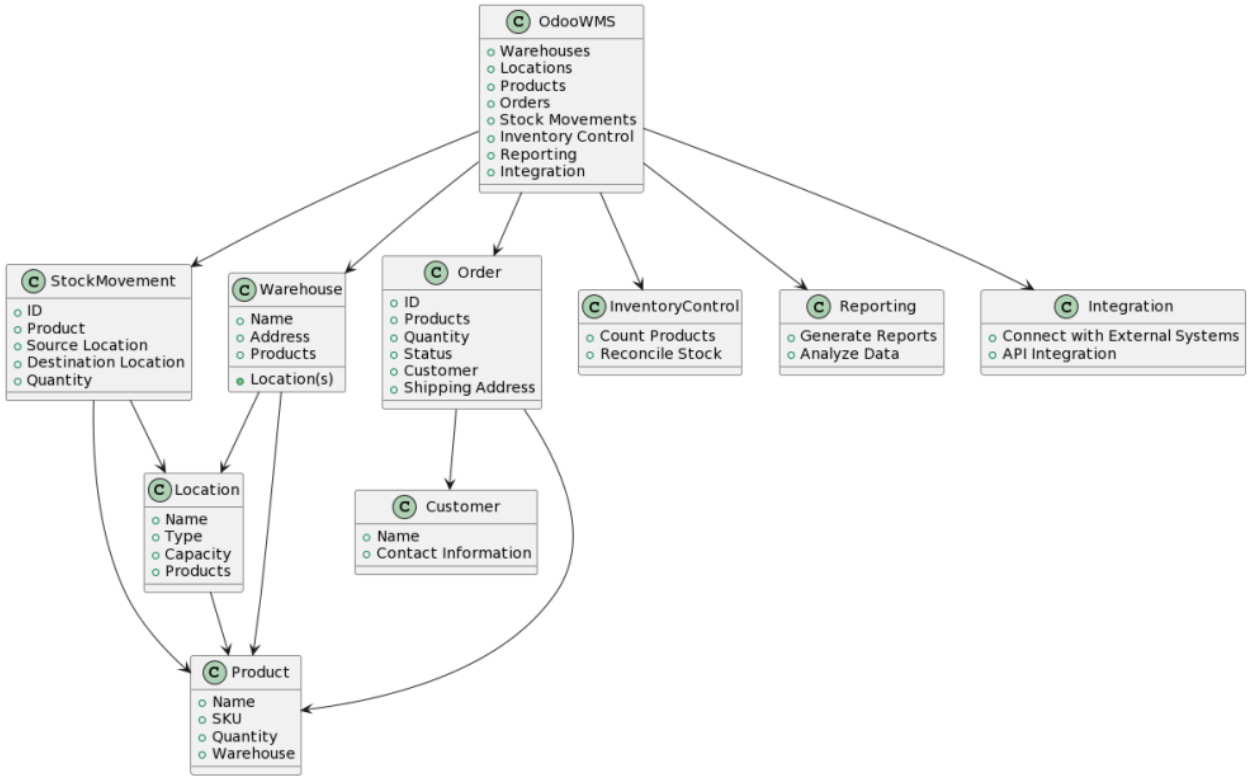


Рисунок 2.9 – Функціональна схема (діаграма класів)

Таким чином, один StockPicking (напр. відвантаження) може включати декілька StockMove – за кількістю різних товарних позицій чи партій, що переміщуються. Для детального відображення фактичного виконання операцій використовується клас StockMoveLine – він фіксує кожну елементарну дію під час переміщення товару (наприклад, відбір конкретної упаковки товару зі стелажа) і пов’язаний зі своїм StockMove. Клас StockMoveLine містить атрибути фактичного переміщення: фактично оброблену кількість одиниць, фактичні локації «звідки-» і «куди-» товар переміщено, часові позначки тощо.

Важливим елементом системи є клас Divergence – спеціальна модель для реєстрації та обробки розбіжностей. Об’єкт Divergence створюється у випадку, коли при виконанні операції виникла невідповідність між плановими параметрами руху товару та фактичними результатами (наприклад, було

заплановано відвантажити 100 одиниць, а фактично відвантажено 95). Модель Divergence пов’язана зі StockMove, який породив розбіжність, і містить дані про тип невідповідності (нестача, надлишок), величину різниці, час фіксації та відповідальну особу.

Окрім названих, до складу діаграми класів входять об’єкти, що представляють довідникову інформацію:

– Product (товарна номенклатура, містить дані про найменування, артикул, характеристики товару);

– Location (складська локація або зона зберігання, має атрибути типу, місткості тощо);

– Warehouse (логістичний центр або склад в цілому, що може об’єднувати кілька локацій);

– Partner/Customer (контрагент – постачальник чи клієнт, від якого надходить або якому відвантажується продукція, зв’язаний з відповідним складським документом).

Зв’язки між класами відображають реальні взаємозалежності: товар (Product) може зберігатися в різних локаціях, кожна операція StockMove має визначені місця відправлення і призначення (Location), декілька StockMove об’єднуються в одному StockPicking, а кожен StockMove може містити одну чи кілька позицій StockMoveLine (якщо операція виконується частинами).

Взаємодія цих класів забезпечує повний цикл обробки складських процесів – від формування замовлення на переміщення чи відвантаження, через резервування запасів і виконання фізичних операцій, до оновлення залишків та формування звітності. Таким чином, функціональна схема задає об’єктно-орієнтовану модель системи, необхідну для подальшої реалізації її бізнес-логіки в коді.

2.8 Алгоритм роботи програмного забезпечення (принципова схема)

Алгоритмічна (принципова) схема (рис. 2.10) відображає послідовність кроків, які виконує система під час проведення типових складських операцій – на прикладі процесу відвантаження товару клієнту.

Рисунок 2.10 – Алгоритм роботи програмного забезпечення (принципова схема)

На початку процесу менеджер або система формує складське замовлення (документ відвантаження) в Odoo, що містить перелік товарів та кількість, яку слід відпустити зі складу. Після цього завдання передається на виконання складу: оператор складу отримує через інтерфейс систему або роздруковане завдання перелік позицій для відбору. Далі оператор використовує ТЗД, щоб послідовно сканувати штрих-коди товарів при їх відборі.

Під час сканування система автоматично створює записи StockMoveLine для кожної знайденої одиниці товару, фіксуючи фактичну кількість і місце розташування. За лаштунками кожне сканування оновлює стан відповідного StockMove (операції) – зменшуючи кількість, що залишилась для відбору, та відмічаючи прогрес виконання.

Коли всі заплановані позиції опрацьовано (або навіть якщо опрацьовано не всі), оператор ініціює завершення операції в системі, тобто підтверджує виконання відвантаження. Це викликає виконання ключових методів Odoo (наприклад, методу _action_done для об’єктів StockMove), які відповідають за фіналізацію транзакції. Під час завершення система автоматично порівнює планову кількість кожного товару з фактично відвантаженою.

Якщо для будь-якої позиції виявлено невідповідність (наприклад, не вистачило товару або був відправлений зайвий товар), спрацьовує механізм обробки розбіжностей – формується об’єкт Divergence, де фіксується вид розбіжності та її величина. Одночасно система коригує дані обліку: зі складу списується фактично відвантажена кількість товару, а різниця (нестача або надлишок) може бути оприбуткована на окремий рахунок або призначена до дозамовлення. Після успішного завершення всіх рухів система переводить документ StockPicking (замовлення на відвантаження) у статус завершеного, генерує необхідні документи (видаткові накладні, акти) та оновлює залишки на складі.

На цьому алгоритм завершується – інформація про виконану операцію зберігається в базі даних і доступна для аналізу. Таким чином, розроблений алгоритм роботи ПЗ гарантує, що всі етапи складської операції автоматизовані

та контрольовані: від формування завдання, через виконання з використанням ТЗД, до завершення з перевіркою та документуванням результатів. Це забезпечує оперативність виконання складських процесів, їхню прозорість і точність обліку, оскільки будь-які відхилення негайно виявляються системою та відображаються для подальшого реагування менеджменту.

2.9 Висновки до розділу

У другому розділі було здійснено вибір та обґрунтування технологій і середовища розробки системи автоматизації логістичного центру, а також розроблено детальну архітектуру майбутньої WMS-підсистеми. На основі результатів розділу остаточно підтверджено вибір ERP-платформи Odoo як ядра системи; наведено обґрунтування цього вибору.

Розглянуто архітектурні особливості Odoo, зокрема її трирівневу клієнт-серверну архітектуру та модульну систему. Описано засоби розробки: мову програмування Python для серверної логіки модулів, механізм опису даних і інтерфейсів через XML та JavaScript-фреймворк OWL для реалізації клієнтської частини.

У цьому ж розділі представлено проектні рішення: розроблено структурну схему системи, що відображає ключові компоненти (база даних, сервер Odoo, клієнти-користувачі і термінали збору даних) та взаємодію між ними.

Крім того, розроблено функціональну схему (діаграму класів), яка деталізує структуру програмної реалізації: визначено основні класи модулю (StockPicking, StockMove, StockMoveLine, Divergence та інші) і їх роль у системі. Також у розділі розроблено алгоритм роботи WMS-підсистеми, який описує послідовність дій при здійсненні складських операцій – від створення документа до оновлення залишків. Цей алгоритм ліг в основу реалізації бізнес-логіки модулю і врахував необхідність контролю розбіжностей та інтеграції з ТЗД.

3 РОЗРОБКА ТА ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Ініціалізація та налаштування проєкту

3.1.1 Ініціалізація та початкове налаштування середовища Odoo

Початковий етап розробки програмного забезпечення передбачає підготовку робочого середовища та встановлення всіх необхідних програмних компонентів, що забезпечують коректне функціонування платформи Odoo.

Першочергово здійснюється інсталяція інтегрованого середовища розробки PyCharm, яке використовується для написання, налагодження та тестування програмного коду мовою Python. Для цього доцільно скористатися офіційним веб-ресурсом компанії JetBrains, де доступна безкоштовна версія PyCharm Community Edition, достатня для реалізації поставлених завдань.

Наступним кроком є встановлення системи управління базами даних PostgreSQL, яка в архітектурі Odoo виконує роль основного сховища даних. Дистрибутив PostgreSQL завантажується з офіційного сайту розробника з урахуванням використовуваної операційної системи. Після інсталяції створюється відповідна база даних та користувач, що надалі використовуються сервером Odoo.

Для розгортання самої платформи необхідно отримати вихідний код Odoo з офіційного репозиторію GitHub. У межах даного проєкту використовується версія Odoo 14.0 Community Edition. Завантаження може бути виконане шляхом клонування репозиторію за допомогою системи контролю версій Git або шляхом завантаження архіву з подальшим його розпакуванням у робочий каталог.

Після отримання вихідного коду обов’язковим є встановлення програмних залежностей, необхідних для коректної роботи Odoo. Для цього у командному рядку виконується команда:

pip install -r requirements.txt

Дана операція забезпечує інсталяцію всіх бібліотек Python, які використовуються серверною частиною платформи.

Для запуску локального сервера Odoo у налаштуваннях конфігурації проєкту необхідно вказати параметр запуску з ключем -c, який посилається на файл конфігурації (рис. 3.1). У цьому файлі задаються параметри підключення до бази даних, порти, шляхи до додаткових модулів та інші системні налаштування.

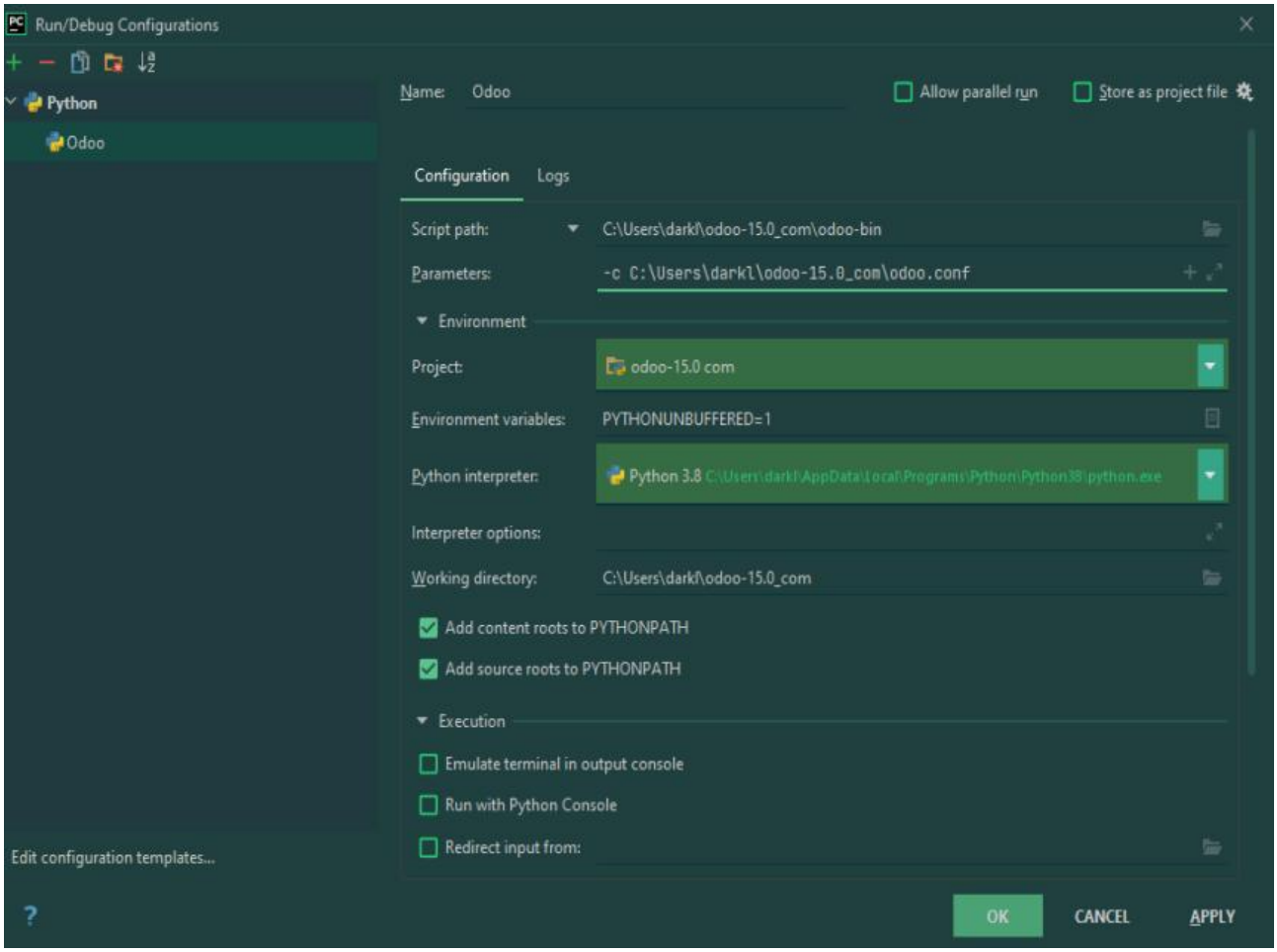


Рисунок 3.1 – Приклад конфігурації проєкту

У разі коректного виконання всіх етапів інсталяції та налаштування, під час запуску сервера у консолі з’являються службові повідомлення, що свідчать про успішне ініціювання серверної частини Odoo та готовність системи до подальшої розробки і конфігурації (рис. 3.2).

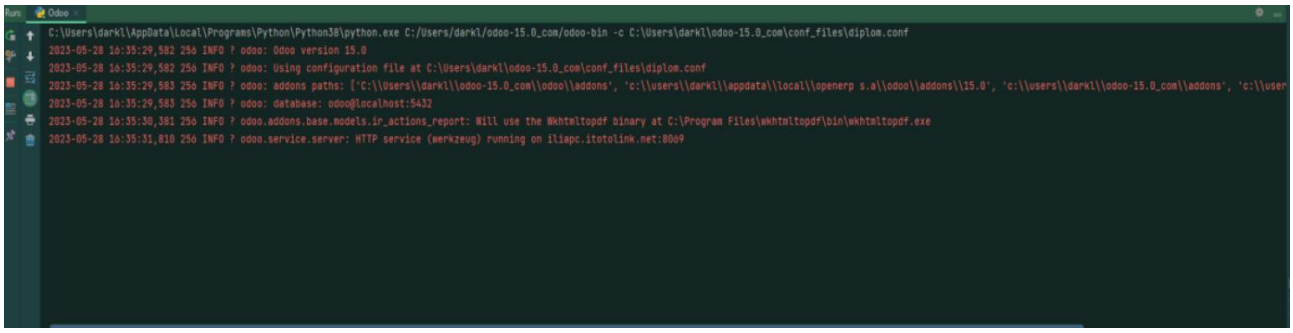


Рисунок 3.2 – Приклад консолі при правильному налаштуванні

Далі потрібно перейти на http://localhost:8069/web після чого буде доступне вікно налаштування бази даних (рис. 3.3).

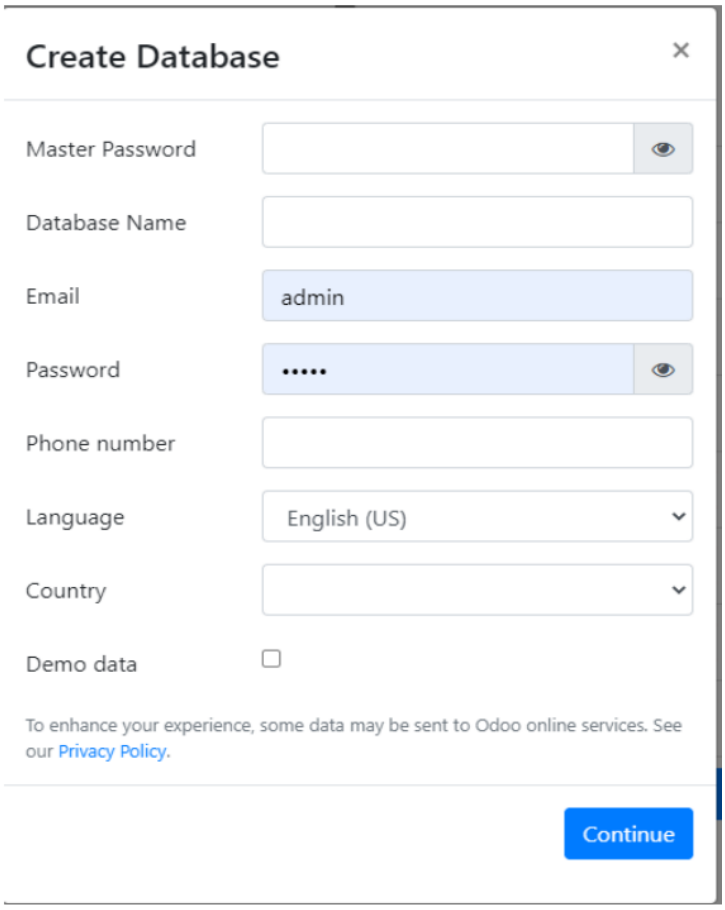


Рисунок 3.3 – Приклад створення нової БД

Після створення нової бази даних та входу в неї ми побачимо вікно доступних модулів (рис. 3.4). Встановлюємо будь-який модуль, наприклад Inventory (склад).

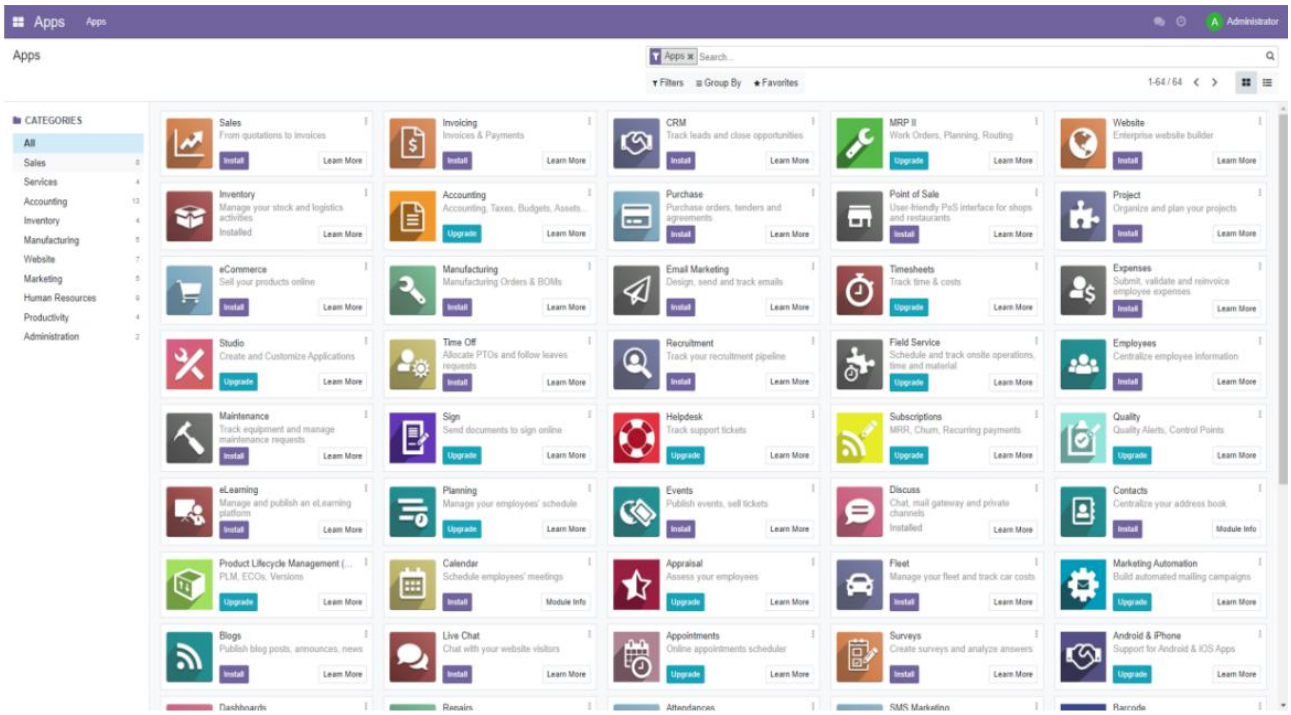


Рисунок 3.4 – Сторінка модулів

Після цього налаштування та встановлення базового варіанту Odoo є завершеним.

3.1.2 Створення та початкове налаштування користувацького модуля

Наступним кроком у процесі розробки є формування власного програмного модуля (рис. 3.5), призначеного для розширення стандартного функціоналу модуля Inventory (складського обліку) в системі Odoo. Реалізація такого модуля дозволяє адаптувати базову WMS-логіку до специфічних вимог проєкту.

Ініціалізація модуля починається зі створення окремого каталогу у директорії додаткових модулів Odoo. У межах даної роботи цей каталог має назву diplom_wms. У створеній директорії формується базова структура файлів і підкаталогів, необхідних для коректної реєстрації та роботи модуля в системі.

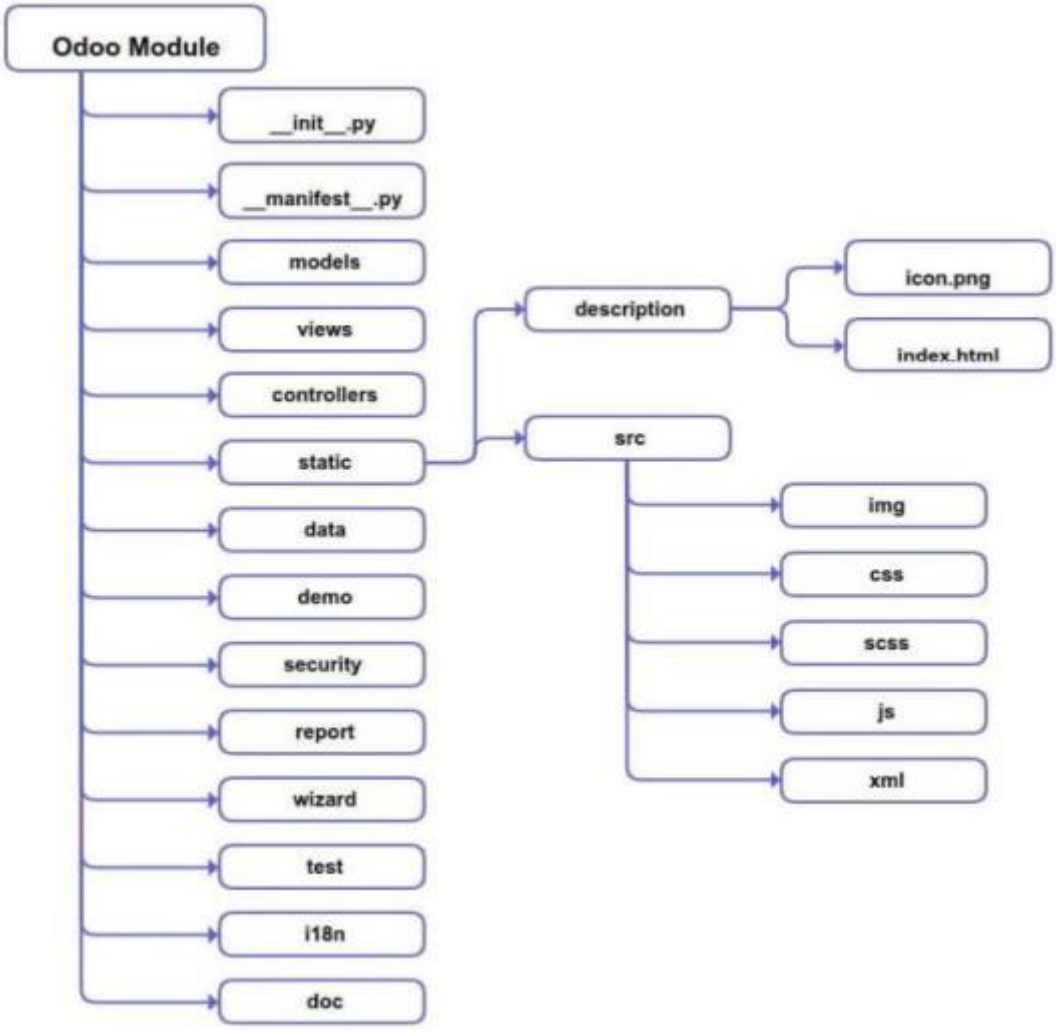


Рисунок 3.5 – Структура модулю в Odoo

До обов’язкових елементів структури модуля належать:

– *__init__.py* – службовий файл, який використовується для ініціалізації модуля як Python-пакета. Як правило, він не містить логіки або містить імпорт основних компонентів модуля;

– *__manifest__.py* – ключовий конфігураційний файл, що описує модуль. У ньому задається службова інформація, зокрема назва модуля, версія, автор, короткий опис призначення, а також перелік залежностей від інших модулів Odoo. Крім того, у маніфесті можуть бути вказані ресурси інтерфейсу, файли перекладу, стилі оформлення та інші допоміжні компоненти;

– *models* – каталог, у якому визначається серверна логіка модуля. Тут створюються Python-файли з описом моделей даних. Можлива як реалізація

нових моделей, так і розширення існуючих моделей базового складського модуля. У моделях описуються поля, методи обробки даних, обчислювані атрибути та механізми реакції на події;

– *views* – директорія, призначена для зберігання XML-файлів, що описують користувацький інтерфейс модуля. У цих файлах визначаються форми введення, спискові подання, меню, дії та інші візуальні елементи, через які користувач взаємодіє з функціоналом модуля.

За необхідності структура модуля може бути доповнена іншими каталогами, зокрема для зберігання файлів керування доступом, JavaScript-скриптів, стилів CSS або ресурсів локалізації.

Окрему увагу при розробці модуля приділяють механізмам безпеки. Файли безпеки в Odoo відіграють важливу роль у контролі доступу до даних і функціональних можливостей системи, особливо в умовах використання інформаційних ресурсів у навчальному або корпоративному середовищі. Вони визначають правила доступу користувачів до моделей, записів і дій у системі.

Основною метою впровадження засобів безпеки є забезпечення конфіденційності, цілісності та доступності даних. Коректне налаштування ролей, прав доступу та обмежень дозволяє мінімізувати ризики несанкціонованого використання інформації, втрати даних або порушення їх цілісності.

Після формування базової структури модуля необхідно підготувати файл *__manifest__.py*, (рис. 3.6) який виконує роль центрального конфігураційного опису модуля в середовищі Odoo. Саме цей файл використовується системою для ідентифікації модуля, визначення його призначення, залежностей та параметрів встановлення.

У файлі *__manifest__.py* задається сукупність атрибутів, що описують функціональне та логічне позиціонування модуля в екосистемі Odoo. Основні параметри, які рекомендовано заповнювати, наведені нижче:

– name – унікальна назва модуля, яка відображається в інтерфейсі системи та використовується для його ідентифікації;

– version – номер версії модуля, що дозволяє відстежувати зміни та оновлення в процесі розвитку програмного продукту;

– author – інформація про розробника або організацію, відповідальну за створення модуля;

– summary – стислий опис призначення модуля, який відображає його основну ідею або ключову функцію;

– description – розширений текстовий опис, у якому деталізується функціональність модуля, його особливості та сфера застосування;

– category – логічна категорія, до якої належить модуль (наприклад, Warehouse, Inventory), що полегшує навігацію та систематизацію модулів у середовищі Odoo;

– depends – перелік інших модулів, від яких залежить робота даного модуля. Наявність цього поля забезпечує коректний порядок встановлення та гарантує доступ до необхідної базової функціональності;

– data – список файлів ресурсів, які підключаються під час встановлення модуля. Сюди можуть входити XML-файли інтерфейсу, файли локалізації, початкові дані для заповнення бази тощо;

– installable – логічний параметр, що визначає можливість інсталяції модуля в системі;

– application – ознака того, чи розглядається модуль як повноцінний прикладний додаток, який може бути відображений у головному меню Odoo;

Перелічені поля формують базовий мінімум конфігурації модуля. За потреби структура файлу *__manifest__.py* може бути розширена додатковими параметрами або змінена відповідно до специфіки реалізованої функціональності та вимог конкретного проєкту.

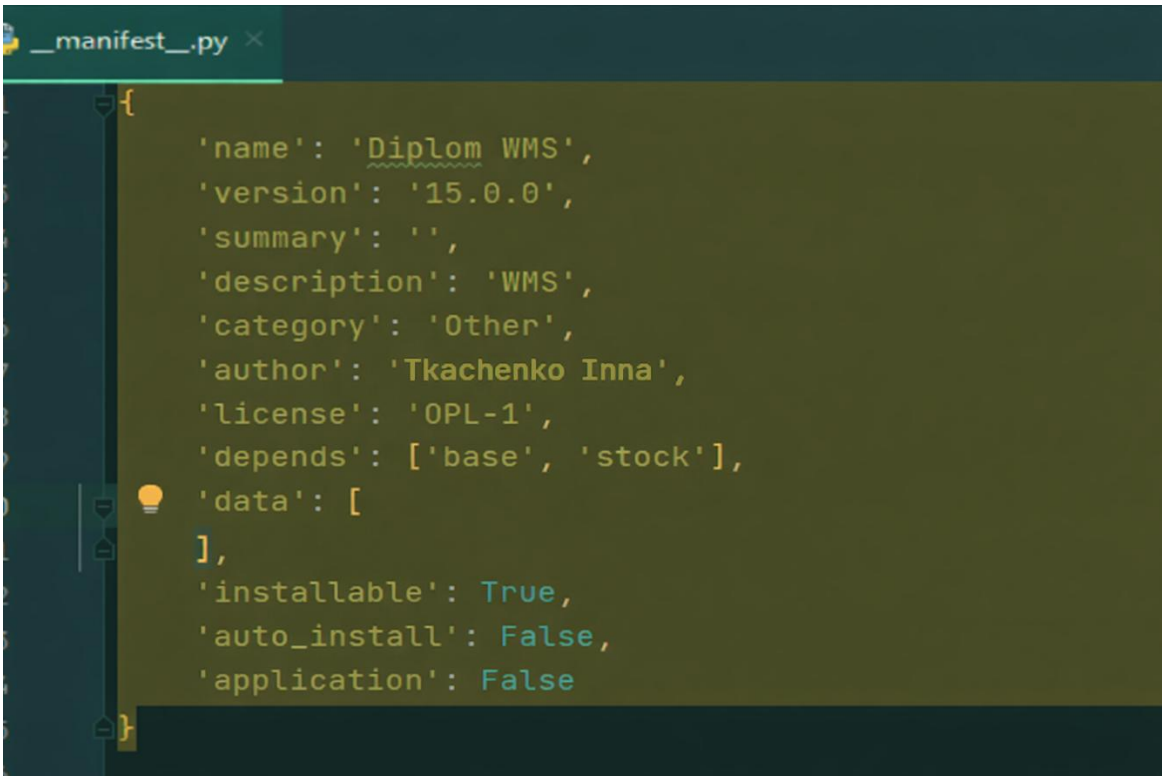


Рисунок 3.6 – Приклад налаштування «*__manifest__.py*»

Для того щоб розроблений користувацький модуль став доступним у середовищі Odoo, необхідно попередньо повідомити систему про місце його розташування. З цією метою у конфігураційному файлі Odoo задається шлях до каталогу, в якому збережено модуль (рис. 3.7). Без виконання цієї дії платформа не зможе виконати індексацію нового розширення.

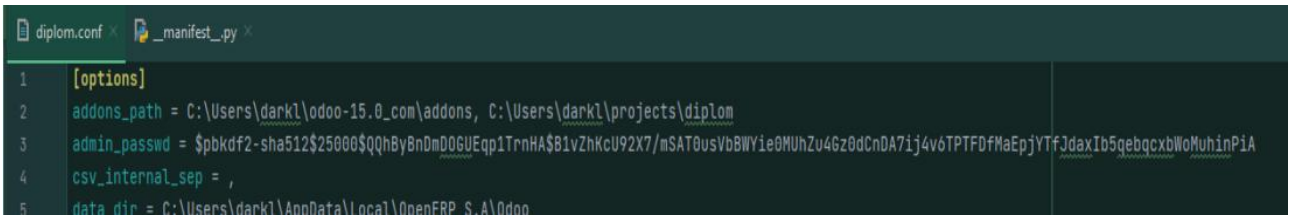


Рисунок 3.7 – Приклад файлу конфігурації

Після внесення відповідних змін до конфігурації та повторного запуску сервера Odoo потрібно ініціювати оновлення переліку модулів у веб-інтерфейсі системи. Для цього виконується наступна послідовність дій:

– відкрити розділ керування модулями, доступний через пункт меню «Apps»;

– запустити процедуру оновлення переліку модулів, скориставшись функцією «Update Apps List», що забезпечує повторне сканування всіх підключених директорій;

– скористатися вбудованим пошуком для знаходження модуля за його назвою;

– активувати встановлення модуля, натиснувши кнопку «Install» (рис. 3.8).

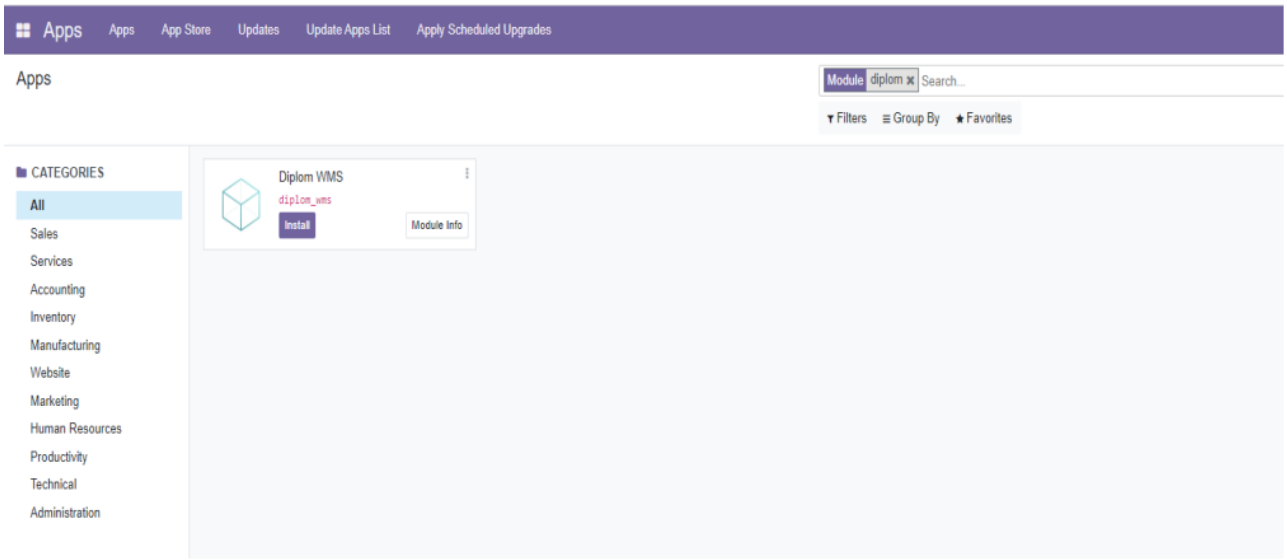


Рисунок 3.8 – Приклад відображення власного модуля

Після завершення зазначених операцій модуль інтегрується в систему Odoo, реєструється в базі даних і стає доступним для подальшого використання та налаштування в межах бізнес-процесів.

3.2 Реалізація back-end частини

3.2.1 Клас StockPicking

У підсистемі складського обліку Odoo модель StockPicking виконує роль базового об’єкта, через який реалізується керування логістичними операціями.

Саме ця модель використовується для фіксації, обробки та контролю всіх рухів матеріальних цінностей у межах складу, незалежно від напряму їх переміщення.

Функціонально StockPicking охоплює повний спектр складських дій:

– надходження товарів від постачальників;

– відвантаження клієнтам;

– внутрішні переміщення між зонами зберігання;

– допоміжні операції, що виникають у процесі управління складом.

Завдяки цьому реалізується єдиний механізм управління складськими документами та їх життєвим циклом.

У стандартній реалізації клас містить інструменти автоматизації відвантажувальних процесів, зокрема:

– формування комплектувальних відомостей;

– резервування товарних позицій;

– списання із зарезервованих залишків;

– фіксацію факту відправлення.

Додатково забезпечується контроль етапів виконання операцій, включно з підтвердженням приймання та завершенням доставки.

Модель StockPicking тісно взаємодіє з іншими компонентами екосистеми Odoo, що дозволяє інтегрувати складські процеси в загальну бізнес-логіку підприємства. Зокрема, вона пов’язана з модулем керування складом, який відповідає за опис конфігурації складів, ієрархію локацій, правила зберігання та організацію внутрішньої структури складського простору.

Таким чином, StockPicking виступає ядром складської підсистеми, забезпечуючи централізований контроль руху товарів, підвищення прозорості операцій та точність облікових даних. Його використання дозволяє оптимізувати логістичні процеси та зменшити кількість помилок, пов’язаних з ручною обробкою інформації.

З метою адаптації системи до вимог повноцінної WMS та інтеграції з терміналами збору даних, стандартний функціонал StockPicking потребує

розширення. Для цього до моделі додаються нові атрибути та логічні обробники (рис. 3.9).

```
divergences_ids = fields.One2many('mo.divergence', 'picking_id')
volume = fields.Float('Volume', compute='_cal_volume', digits='Volume')
volume_actual = fields.Float('Volume Actual', compute='_cal_val_actual', digits='Volume')
weight_actual = fields.Float('Weight Actual', compute='_cal_val_actual', digits='Stock Weight')
```

Рисунок 3.9 – Додавання нових полів

Зокрема, вводиться поле «wms_state», яке використовується для відображення стану складського документа під час роботи з мобільними ТЗД. Також реалізується зв’язок «divergences_ids», що дозволяє фіксувати розбіжності між плановими та фактичними даними. Окрему групу становлять поля, призначені для зберігання параметрів вантажу, а саме фактичного та розрахункового об’єму й маси: «volume», «volume_actual», «weight_actual».

3.2.2 Клас Divergence

Модель Divergence, що функціонально пов’язана з об’єктом StockMove, призначена для фіксації та обробки невідповідностей, які виникають між плановими параметрами товарного руху та фактичними результатами складських операцій (рис. 3.10). Її застосування спрямоване на підвищення достовірності обліку та стабільності системи управління запасами.

Основне призначення Divergence полягає у виконанні аналітичного зіставлення очікуваних і реальних показників. У межах цього процесу здійснюється перевірка таких атрибутів, як замовлений обсяг продукції, її вартісні характеристики, заплановані та фактичні дати переміщення, а також реальна кількість отриманих товарних одиниць. Джерелом цих даних виступають записи, сформовані в моделі StockMove.

У випадку виявлення відхилень між розрахунковими та фактичними значеннями система автоматично формує окремий запис розбіжності, який

прив’язується до відповідного документа складського переміщення. Таким чином забезпечується прозора фіксація помилок, нестач, надлишків або інших порушень у ланцюгу постачання.

```python
class Divergence(models.Model):
    _name = 'mo.divergence'
    _description = 'Divergence in stock picking'

    product_id = fields.Many2one('product.product', string='Product')
    deviation = fields.Float('Deviation')
    demand = fields.Float('Demand')
    picking_id = fields.Many2one('stock.picking', ondelete='cascade')
    move_id = fields.Many2one('stock.move', ondelete='cascade')
```

Рисунок 3.10 – Клас Divergence

Після реєстрації розбіжностей модель Divergence може ініціювати набір подальших дій: формування службових повідомлень, ведення журналів подій, інформування відповідальних користувачів, коригування даних у базі або запуск автоматизованих процедур з усунення виявлених невідповідностей (рис. 3.11).

```python
def update_divergences(self):
    # create deviation logs
    if self.divergences_ids:
        self.divergences_ids.unlink()
    for line in self.move_ids_without_package:
        if line.picking_id.picking_type_code == 'internal':
            if line.reserved_availability != line.quantity_done:
                self.divergences_ids.create({
                    'product_id': line.product_id.id,
                    'deviation': line.deviation,
                    'picking_id': line.picking_id.id,
                    'demand': line.product_uom_qty,
                    'move_id': line.id
                })
        else:
            if line.deviation > 0 or line.deviation < 0:
                self.divergences_ids.create({
                    'product_id': line.product_id.id,
                    'deviation': line.deviation,
                    'picking_id': line.picking_id.id,
                    'demand': line.product_uom_qty,
                    'move_id': line.id
                })
```

Рисунок 3.11 – Приклад функції для створення логів при виявлені розбіжності

Загалом, використання класу Divergence є критично важливим для забезпечення контрольованості складських процесів. Його впровадження дозволяє своєчасно ідентифікувати проблемні ситуації, мінімізувати втрати, підвищити точність управління запасами та, як наслідок, покращити рівень сервісу для кінцевих споживачів.

3.2.3 Клас StockMove

Клас StockMove в системі Odoo виконує роль базового облікового елемента, що описує одиничний акт переміщення матеріальних цінностей у межах складської інфраструктури. Кожен об’єкт цього класу відповідає конкретній операції руху товару між визначеними локаціями та є первинною одиницею деталізації складських процесів.

У межах моделі StockMove акумулюється ключова інформація, необхідна для точного контролю руху запасів, зокрема: ідентифікатор операції, номенклатурна позиція товару, кількісні показники, початкове та кінцеве місце зберігання, а також поточний стан виконання операції. Наявність цих атрибутів дозволяє системі забезпечувати коректний облік, прозорість логістичних процесів та оперативний контроль за складськими залишками.

Функціонально StockMove охоплює повний спектр операцій, пов’язаних із товарообігом, включаючи надходження продукції на склад, відвантаження клієнтам, внутрішні переміщення між зонами зберігання, а також операції повернення. Для кожного переміщення передбачено механізм відстеження стану виконання, що дозволяє контролювати послідовність дій та запобігати помилкам у процесі обробки товарів.

Модель StockMove тісно інтегрована з іншими компонентами Odoo, зокрема з класом StockPicking, що забезпечує агрегування окремих переміщень у логічно завершені складські документи. Завдяки такій взаємодії реалізується безперервний обмін даними між підсистемами, що охоплює формування замовлень, резервування запасів, актуалізацію залишків, а також підтвердження факту виконання операцій.

У рамках даної роботи клас StockMove було доопрацьовано з урахуванням вимог WMS-підходу та інтеграції з терміналами збору даних (ТЗД). Зокрема, модифіковано окремі методи, серед яких ключовим є _action_done, що забезпечує коректне завершення операцій переміщення з урахуванням фактичних даних, отриманих у процесі сканування та виконання складських дій (рис. 3.12).

```
def _action_done(self, cancel_backorder=False):
    res = super()._action_done(cancel_backorder)
    if all(move.state == 'done' for move in self.picking_id):
        self.picking_id.write({
            'wms_state': 'done'
        })
    return res
```

Рисунок 3.12 – Приклад оновлення функції «_action_done»

3.2.4 Клас StockMoveLine

Клас StockMoveLine у складі WMS-підсистеми Odoo виконує функцію детального опису фактичних операцій руху товарів, фіксуючи кожну елементарну дію, що відбувається під час складських переміщень. На відміну від узагальнених облікових моделей, цей клас орієнтований на операційний рівень та відображає реальний перебіг виконання складських завдань.

Кожен об’єкт StockMoveLine інтерпретується як окрема операційна позиція, у якій зосереджено повний набір параметрів, необхідних для коректного виконання та контролю переміщення: номенклатура товару, фактична кількість, вихідна та цільова локації, ідентифікатор операції, а також стан виконання. Така деталізація забезпечує можливість точного простежування руху матеріальних ресурсів у межах складу.

Функціональне призначення StockMoveLine полягає у забезпеченні точності складських операцій за рахунок контролю кількісних показників,

фіксації маршруту переміщення та актуалізації статусів виконання. Саме цей клас слугує основою для перевірки відповідності між запланованими діями та їх фактичним виконанням у WMS-системі.

У межах розробленої підсистеми клас StockMoveLine було модифіковано з урахуванням інтеграції з терміналами збору даних (ТЗД). Зокрема, переглянуто та адаптовано ключові методи, що відповідають за введення, обробку та підтвердження інформації, отриманої безпосередньо під час виконання складських операцій.

Розширення функціональності передбачало додавання нових механізмів взаємодії з ТЗД, які забезпечують безперервний обмін даними у режимі реального часу (рис. 3.13). У результаті реалізовано підтримку сканування штрих-кодів, введення фактичних кількостей товару, фіксацію завершення операцій та оперативне оновлення станів переміщень у системі.

Застосовані зміни дозволили суттєво підвищити достовірність складських даних, скоротити час виконання операцій та мінімізувати ймовірність помилок, пов’язаних із ручним введенням інформації, що безпосередньо вплинуло на ефективність роботи складу.

```python
@api.model_create_multi
def create(self, vals_list):
    line_picking = []
    for vals in vals_list:
        if 'qty_done' in vals and vals.get("picking_id", False):
            line_picking.append(vals['picking_id'])
            break

    if line_picking:
        self.env['stock.picking'].browse(line_picking).write({
            'wms_state': 'in_work'
        })

    res = super().create(vals_list)
    return res
```

Рисунок 3.13 – Перероблення функції «create»

3.2.5 Клас ProductProduct

Клас ProductProduct у складі WMS-підсистеми Odoo виконує роль базової облікової моделі, що централізовано описує номенклатуру складських ресурсів та забезпечує керування інформацією про кожну товарну позицію. Саме через цей клас формується єдине джерело достовірних даних щодо характеристик продукції, яка обробляється складською системою.

Кожен об’єкт ProductProduct інтерпретується як окрема одиниця товару з унікальним набором параметрів, що визначають його ідентичність та властивості. До таких параметрів належать комерційні та облікові характеристики, зокрема найменування, внутрішні коди, опис, ціна, одиниці обліку, категоріальна належність, а також поточний стан доступності. Сукупність цих даних забезпечує коректну ідентифікацію товару в усіх складських процесах.

Функціональне призначення класу полягає у підтримці актуального стану запасів, що включає контроль кількості наявних одиниць, їх розміщення у складських локаціях та статуси використання (доступний, зарезервований, заблокований, у процесі замовлення тощо). Це створює передумови для ефективного планування, аналізу та управління матеріальними потоками.

Клас ProductProduct тісно інтегрований з іншими складськими моделями Odoo, зокрема зі StockMove та StockMoveLine, що дозволяє автоматизувати повний цикл операцій з товаром – від формування замовлень і резервування до фізичного переміщення, відвантаження та повернення. Така взаємодія забезпечує цілісність даних і узгодженість дій у межах WMS.

У межах розширення стандартного функціоналу WMS до моделі ProductProduct були додані додаткові атрибути, що описують геометричні параметри товару, а саме висоту, ширину та довжину (рис. 3.14). Введення цих характеристик дозволяє перейти від абстрактного обліку номенклатури до урахування реальних фізичних властивостей продукції.

```
class Product(models.Model):
    _inherit = "product.product"

    height = fields.Float('Height', digits='Size')
    width = fields.Float('Width', digits='Size')
    depth = fields.Float('Depth', digits='Size')
```

Рисунок 3.14 – Нові поля у класі ProductProduct

Використання просторових параметрів товару є критично важливим для оптимізації складської логістики, зокрема для раціонального використання складського простору, планування розміщення продукції в палетах або контейнерах, а також для розрахунку логістичних витрат, що залежать від об’ємних характеристик вантажу.

3.2.6 Клас StockQuant

Клас StockQuant у системі Odoo виконує функцію базової одиниці кількісного обліку складських залишків і є ключовим елементом механізму контролю запасів. Саме через цю модель реалізується фіксація фактичної наявності товарів у конкретних складських локаціях у реальному часі.

Кожен об’єкт StockQuant відображає поєднання товарної позиції та місця її зберігання із зазначенням точної кількості одиниць. Такий підхід дозволяє деталізувати складський облік до рівня окремих комірок, зон або адрес зберігання, забезпечуючи високу точність даних щодо розподілу матеріальних ресурсів.

Функціональність класу передбачає автоматичне коригування залишків у відповідь на всі типи складських операцій, зокрема приймання продукції, відвантаження, внутрішні переміщення, повернення та інші дії, що впливають на стан запасів. На основі цих даних система визначає фактично доступну кількість товару, відслідковує динаміку змін та забезпечує узгодженість виконання складських процесів.

У межах розширення стандартних можливостей WMS до моделі StockQuant було введено додатковий атрибут та відповідну обчислювальну функцію для визначення об’єму, зайнятого товаром у конкретній складській комірці – quant_volume (рис. 3.15). Реалізація цього показника дозволяє перейти від суто кількісного обліку до просторового аналізу заповнення складу.

```
quant_volume = fields.Float('Quant Volume', compute='_compute_volume', store=True, digits='Volume')

@api.depends('available_quantity')
def _compute_volume(self):
    for item in self:
        item.quant_volume = item.product_id.volume * item.available_quantity
```

Рисунок 3.15 – Поле та функція для розрахунку зайнятого об’єму комірки

Застосування параметра зайнятого об’єму створює передумови для ефективного управління складським простором, контролю перевантаження комірок, оптимізації розміщення товарів та підвищення загальної точності планування логістичних операцій.

3.2.7 Клас StockInventory

Клас StockInventory у складі WMS-підсистеми Odoo призначений для реалізації процедур контролю фактичних залишків товарів та підтримання актуальності складських даних. Його основне завдання полягає у забезпеченні відповідності між реальним станом запасів і обліковою інформацією, що зберігається в системі.

Функціональні можливості цієї моделі охоплюють організацію та проведення планових і позапланових інвентаризацій, під час яких виконується фізичний підрахунок наявних товарів із подальшим зіставленням отриманих результатів із системними значеннями. Такий підхід дозволяє своєчасно виявляти відхилення, що виникають у процесі експлуатації складу, та ініціювати їх коригування.

Модель StockInventory забезпечує формування інвентаризаційних документів, вибір зон або позицій зберігання, фіксацію фактичних кількостей, автоматичне внесення корекцій та оновлення залишків у базі даних. Додатково реалізовано інструменти аналізу виявлених розбіжностей і генерації звітних матеріалів, що спрощує роботу персоналу та підвищує прозорість інвентаризаційних процесів (рис. 3.16).

```
class Inventory(models.Model):
    _inherit = "stock.inventory"

    wms_state = fields.Selection([('draft', 'Draft'),
                                  ('in_work', 'In Work'),
                                  ('done', 'Done'),
                                  ('cancel', 'Canceled')], string="WMS State", default='draft', tracking=True,
                                 copy=False)

    def action_cancel_draft(self):
        super().action_cancel_draft()
        self.write({
            "wms_state": 'draft'
        })
```

Рисунок 3.16 – Приклад нового поля та перероблення функції «action_cancel_draft»

Використання класу StockInventory сприяє підвищенню достовірності складського обліку, посиленню контролю за матеріальними ресурсами та зменшенню ризиків накопичення помилок у даних. Це, у свою чергу, позитивно впливає на ефективність управління запасами та якість прийняття управлінських рішень.

У межах розробленої WMS-системи модель StockInventory була доопрацьована з урахуванням інтеграції з ТЗД. Зміни стосувалися адаптації ключових механізмів інвентаризації та розширення структури даних шляхом додавання нових атрибутів, необхідних для коректної взаємодії з мобільними пристроями.

Реалізовані доопрацювання забезпечили підтримку автоматизованого збору фактичних даних безпосередньо під час інвентаризації, що дозволило скоротити час виконання операцій, зменшити вплив людського фактору та підвищити точність результатів.

3.3 Розроблення front-end частини

3.3.1 Створення нових відображень

Розробка користувацьких відображень у системі Odoo здійснюється з використанням XML-описів, які є основним механізмом формування інтерфейсу та керування його поведінкою. Застосування XML дозволяє формалізовано визначати структуру екранів, логіку відображення даних та взаємодію користувача з елементами системи без безпосереднього втручання в ядро платформи.

У процесі створення нових відображень (рис. 3.17) розробник має можливість описувати різні типи інтерфейсних компонентів, зокрема форми введення, спискові представлення, кнопки дій, поля введення, фільтри та інші елементи керування. Для цього використовуються спеціалізовані XML-теги та атрибути, які визначають розташування елементів, їх доступність, залежність від стану об’єктів і сценарії взаємодії.

Для налаштування зовнішнього вигляду створених відображень у межах XML-файлів можуть застосовуватися стилістичні параметри, що дозволяють керувати кольоровою схемою, розмірами елементів, шрифтами та іншими візуальними характеристиками інтерфейсу. Це забезпечує узгодженість користувацького досвіду та відповідність інтерфейсу корпоративним вимогам.

Окрім статичного опису структури, XML-відображення в Odoo можуть бути тісно інтегровані з серверною логікою. Через механізми зв’язку з Python-кодом реалізується динамічна поведінка інтерфейсу, зокрема перевірка коректності введених даних, обчислення значень, керування доступністю елементів та отримання інформації з бази даних у реальному часі.

Для використання нових відображень у системі необхідно включити відповідні XML-файли до складу модуля та активувати його в середовищі Odoo. Після цього створені інтерфейсні елементи стають доступними в межах відповідних бізнес-процесів і можуть бути використані спільно з іншими модулями системи.

```xml
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <record id="mo_set_arrive_zone_form_view" model="ir.ui.view">
        <field name="name">mo.set.arrive.zone.form</field>
        <field name="model">mo.set.arrive.zone</field>
        <field name="arch" type="xml">
            <form>
                <group>
                    <field name="parent_location_id_comp" invisible="1"/>
                    <field name="arrive_zone" required="1"/>
                    <field name="car_and_trailer_number"/>
                    <field name="ttn"/>
                    <field name="order"/>
                    <field name="parent_location_id" invisible="1"/>
                    <field name="euro_pallet_count_arrive"/>
                    <field name="other_pallet_count_arrive"/>
                </group>
                <footer>
                    <button name="set_arrive_zone" string="Apply" type="object" class="btn-primary"/>
                    <button string="Cancel" class="btn-secondary" special="cancel"/>
                </footer>
            </form>
        </field>
    </record>

    <record id="mo_set_arrive_zone_form_wizard" model="ir.actions.act_window">
        <field name="name">Set Arrive Zone</field>
        <field name="type">ir.actions.act_window</field>
        <field name="res_model">mo.set.arrive.zone</field>
        <field name="view_mode">form</field>
        <field name="view_id" ref="mo_set_arrive_zone_form_view"/>
        <field name="target">new</field>
    </record>

</odoo>
```

Рисунок 3.17 – Створення нового відображення

Таким чином, механізм створення відображень за допомогою XML у Odoo є гнучким та ефективним інструментом адаптації системи до специфічних вимог підприємства. Він дозволяє розширювати та модифікувати функціональність платформи, забезпечуючи зручний, інтуїтивно зрозумілий та продуктивний користувацький інтерфейс.

3.3.2 Доповнення існуючих елементів за допомогою XPath

Одним із ключових механізмів розширення та модифікації інтерфейсу в системі Odoo є використання технології XPath. Вона дозволяє здійснювати точкове доповнення або зміну вже наявних елементів користувацького

інтерфейсу без необхідності повного перевизначення XML-відображень, що істотно спрощує процес кастомізації системи (рис. 3.18).

```
<xpath expr="//page[@name='note']" position="before">
    <page string="Divergences" name="divergences">
        <button name="update_divergences" position="inside" type="object">Update</button>
        <field name="divergences_ids">
            <tree decoration-info="deviation &gt; 0" decoration-danger="deviation &lt; 0">
                <field name="product_id"/>
                <field name="demand"/>
                <field name="deviation"/>
                <field name="picking_id" invisible="1"/>
                <field name="move_id" invisible="1"/>
            </tree>
        </field>
    </page>
</xpath>
```

Рисунок 3.18 – Використання Xpath

XPath надає можливість точно визначати шлях до конкретного елемента в ієрархічній структурі XML-документа. За допомогою XPath-виразів можна здійснювати вибір елементів за їх тегами, атрибутами, значеннями або логічними умовами. Це дозволяє ідентифікувати як окремі компоненти інтерфейсу, так і групи елементів, що підлягають модифікації.

Після визначення цільового елемента з використанням XPath розробник може змінювати його властивості, доповнювати новими атрибутами, редагувати текстові значення або коригувати вкладену структуру. Окрім цього, XPath дозволяє вставляти нові елементи до існуючих блоків, змінювати порядок відображення компонентів або, за необхідності, приховувати окремі елементи інтерфейсу.

Застосування XPath забезпечує гнучке налаштування користувацького інтерфейсу Odoo відповідно до специфіки бізнес-процесів. За його допомогою можна додавати нові кнопки керування, поля введення, інформаційні блоки або розширювати таблиці даних, не порушуючи базову логіку роботи системи та не змінюючи стандартні модулі.

Таким чином, використання XPath у процесі доповнення існуючих елементів інтерфейсу дозволяє ефективно адаптувати Odoo до індивідуальних вимог підприємства. Даний підхід мінімізує обсяг змін у базовому коді, підвищує підтримуваність рішень та забезпечує зручність подальшого розвитку і супроводу програмного забезпечення.

3.4 Розроблення інтерфейсу ТЗД

3.4.1 Інтерфейс ТЗД

Розробка спеціалізованого інтерфейсу для ТЗД з використанням фреймворку Vue.js у середовищі Odoo забезпечує гнучку адаптацію користувацького інтерфейсу до потреб складської логістики та дозволяє оптимізувати виконання операцій у режимі реального часу.

Початковим етапом створення інтерфейсу ТЗД є розробка окремого модуля Odoo, у межах якого визначаються клієнтські компоненти, шаблони відображення та логіка взаємодії з серверною частиною системи. У відповідних файлах модуля описується структура інтерфейсу, правила обробки подій та механізми обміну даними між фронтендом і бекендом.

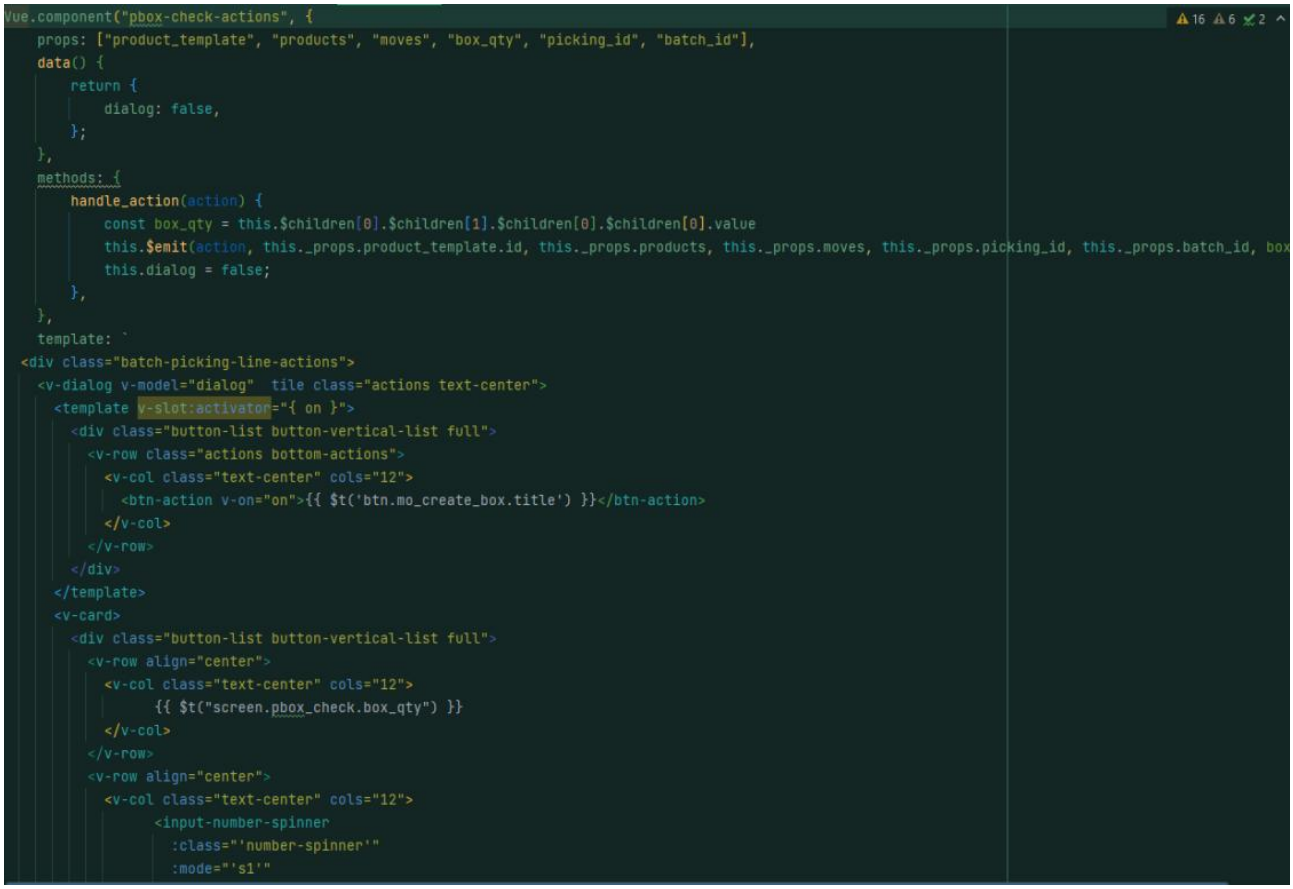
Використання Vue.js дозволяє реалізувати інтерфейс у вигляді незалежних компонентів, таких як форми введення, таблиці, елементи навігації та керуючі кнопки. Шаблонний синтаксис Vue.js забезпечує зв’язування відображуваних даних зі змінними та методами JavaScript, що спрощує динамічне оновлення інформації на екрані ТЗД.

Однією з ключових переваг застосування Vue.js є підтримка двостороннього зв’язування даних, яке забезпечує автоматичну синхронізацію стану інтерфейсу з даними прикладної логіки. Завдяки цьому зміни, ініційовані користувачем під час роботи з ТЗД, миттєво відображаються в інтерфейсі, а оновлення даних з серверної частини не потребує перезавантаження сторінки.

Взаємодія інтерфейсу ТЗД зі складськими операціями реалізується через API-запити до серверної частини Odoo. Такий підхід дозволяє отримувати та

передавати дані, змінювати статуси складських документів, фіксувати результати сканування та виконувати необхідні бізнес-операції безпосередньо з ТЗД.

Додавання нового інтерфейсного компонента на базі Vue.js передбачає створення відповідного файлу, що містить шаблон, сценарій та опис стилів (рис. 3.19). У шаблоні визначається HTML-структура компонента, у сценарії реалізується логіка роботи, включаючи обробку подій, методи доступу до даних і допоміжні обчислення.



```
Vue.component("pbox-check-actions", {
  props: ["product_template", "products", "moves", "box_qty", "picking_id", "batch_id"],
  data() {
    return {
      dialog: false,
    };
  },
  methods: {
    handle_action(action) {
      const box_qty = this.$children[0].$children[1].$children[0].$children[0].value
      this.$emit(action, this._props.product_template.id, this._props.products, this._props.moves, this._props.picking_id, this._props.batch_id, box
      this.dialog = false;
    },
  },
  template: `
<div class="batch-picking-line-actions">
 <v-dialog v-model="dialog"  tile class="actions text-center">
  <template v-slot:activator="{ on }">
   <div class="button-list button-vertical-list full">
    <v-row class="actions bottom-actions">
     <v-col class="text-center" cols="12">
      <btn-action v-on="on">{{ $t('btn.mo_create_box.title') }}</btn-action>
     </v-col>
    </v-row>
   </div>
  </template>
  <v-card>
   <div class="button-list button-vertical-list full">
    <v-row align="center">
     <v-col class="text-center" cols="12">
          {{ $t("screen.pbox_check.box_qty") }}
     </v-col>
    </v-row>
    <v-row align="center">
     <v-col class="text-center" cols="12">
          <input-number-spinner
           :class="'number-spinner'"
           :mode="'s1'"
```

Рисунок 3.19 – Додавання нового компонента в Vue.js

Обробка подій сканування на ТЗД здійснюється через виклик JavaScript-функцій, які ініціюють API-запити до серверної частини. На стороні Odoo визначається Python-контролер, відповідальний за прийом та обробку цих запитів. Контролер виконує аналіз сканованих даних, здійснює пошук у базі

даних, оновлює параметри складських об’єктів або змінює їхній стан відповідно до логіки WMS (рис. 3.20).

```
on_scan: (scanned) => {
    this.wait_call(
        this.odoo.call("scan_line", {
            picking_batch_id: this.current_batch().id,
            move_line_id: this.state.data.id,
            barcode: scanned.text,
        })
    );
},
```

Рисунок 3.20 – Виклик функції при скануванні на ТЗД

Після завершення обробки запиту сервер формує відповідь, як правило, у форматі JSON, яка повертається до клієнтської частини. Отримані дані інтерпретуються фронтендом і використовуються для оновлення відображення інформації на екрані ТЗД, що забезпечує безперервний та інтерактивний процес роботи користувача.

3.4.2 Обробка даних від ТЗД у бекенд-частині

Серверна частина обробки даних від терміналів збору даних у системі Odoo реалізована мовою Python із використанням механізмів контролерів та моделей фреймворку. Саме бекенд відповідає за інтерпретацію подій, що надходять з ТЗД, перевірку коректності отриманої інформації та виконання відповідних складських операцій.

Архітектура обробки побудована на Python-контролерах, які приймають HTTP-запити від клієнтського інтерфейсу ТЗД. Кожен контролер містить набір методів, що відповідають окремим типам подій, ініційованих користувачем під час роботи з терміналом, зокрема сканування штрих-кодів, введення кількості товару або підтвердження виконання операції.

У межах цих методів реалізується прикладна логіка, характерна для WMS-систем. Залежно від типу події, бекенд може виконувати пошук товару за ідентифікатором або штрих-кодом, перевіряти його наявність і доступну кількість у конкретній складській локації, аналізувати поточний стан складського документа та оновлювати статуси відповідних операцій (рис. 3.21).

```
def scan_line(self, picking_batch_id, move_line_id, barcode):
    batch = self.env["stock.picking.batch"].browse(picking_batch_id)
    if not batch.exists():
        return self._response_batch_does_not_exist()
    move_line = self.env["stock.move.line"].browse(move_line_id)
    if not move_line.exists():
        return self._pick_next_line(
            batch, message=self.msg_store.operation_not_found()
        )

    search = self._actions_for("search")

    picking = move_line.picking_id

    package = search.package_from_scan(barcode)
    if package and move_line.package_id == package:
        return self._scan_line_by_package(picking, move_line, package)

    # use the common search method so we search by packaging too
    product = search.product_from_scan(barcode)
    if product and move_line.product_id == product:
        return self._scan_line_by_product(picking, move_line, product)
```

Рисунок 3.21 – Функція для обробки даних ТЗД при скануванні

Отримані від ТЗД дані проходять перевірку та обробляються з використанням моделей Odoo, що забезпечує цілісність і узгодженість інформації в базі даних. За необхідності бекенд фіксує результати операцій, створює або оновлює записи складських переміщень, а також зберігає додаткові параметри, що були зчитані або введені оператором.

Такий підхід дозволяє централізувати логіку обробки подій від ТЗД, мінімізувати кількість помилок під час складських операцій та забезпечити синхронну взаємодію між клієнтським інтерфейсом і серверною частиною системи. У результаті досягається підвищення точності обліку, стабільності роботи WMS та швидкості виконання операцій у складському середовищі.

3.4.3 Валідація даних від ТЗД у бекенд-частині

Перевірка коректності даних, що надходять від терміналів збору даних, є одним із ключових етапів серверної обробки інформації в системі управління складом. Саме на етапі валідації визначається, чи можуть отримані дані бути безпечно та коректно використані для виконання складських операцій (рис. 3.22).

Валідація забезпечує відповідність інформації, отриманої зі сканерів штрих-кодів та інших пристроїв ТЗД, визначеним вимогам до структури, типів і формату даних. Це дозволяє запобігти потраплянню до WMS некоректних, неповних або логічно помилкових значень, які можуть призвести до порушення обліку запасів або некоректної роботи бізнес-логіки.

У розробленій системі застосовано підхід валідації на основі JSON-схем. Використання схем дозволяє формалізувати вимоги до вхідних даних, зокрема визначити обов’язкові поля, допустимі типи значень, обмеження діапазонів та логічні залежності між параметрами. Таким чином забезпечується автоматизована перевірка відповідності отриманих даних заданому шаблону.

Під час надходження інформації від ТЗД серверна частина системи виконує її первинну перевірку з використанням стандартних засобів мови Python. Дані аналізуються у вигляді структур типу «dict», а також із застосуванням можливостей стандартної бібліотеки «json». На цьому етапі перевіряється коректність формату, наявність необхідних ключів і відповідність значень очікуваним типам.

У разі успішного проходження валідації дані передаються до наступних етапів обробки, де використовуються для оновлення складських документів і

виконання відповідних операцій. Якщо ж виявляються порушення схеми або помилки формату, запит відхиляється, а система може сформувати повідомлення про помилку для клієнтської частини ТЗД.

```
class ShopfloorSchemaAction(Component):

    _inherit = "shopfloor.schema.action"

    def package(self, with_packaging=False):
        schema = super().package(with_packaging=with_packaging)
        schema['shopfloor_closed'] = {"type": "boolean", "required": False, "nullable": True}
        return schema

    def move_lines_counters(self):
        return {
            "lines_count": {"type": "float", "required": False, "nullable": True},
            "packages_count": {"type": "float", "required": False, "nullable": True},
            "use_count_packages": {"type": "boolean", "required": False, "nullable": True},
            "picking_count": {"type": "float", "required": False, "nullable": True},
            "priority_lines_count": {
                "type": "float",
                "required": False,
                "nullable": True,
            },
            "priority_picking_count": {
                "type": "float",
                "required": False,
                "nullable": True,
            },
        }

    def printed_product_box_for_list(self):
```

Рисунок 3.22 – Приклад схем валідації даних

Застосування формалізованої валідації на серверному рівні підвищує надійність WMS, забезпечує цілісність даних та зменшує ймовірність виникнення помилок під час виконання складських процесів у реальному часі.

3.5 Огляд системи

3.5.1 Огляд інтерфейсу ТЗД

Інтерфейс термінала збору даних реалізований у вигляді структурованого меню, яке охоплює базові складські операції в межах WMS-системи. До основних пунктів меню належать операції приймання товару, відвантаження, комплектації, внутрішнього переміщення, а також інші дії, пов’язані з рухом матеріальних цінностей на складі (рис. 3.23).

Вибір конкретної операції ініціює відповідний сценарій роботи, наступним етапом якого є сканування ідентифікаційного об’єкта. Залежно від типу операції таким об’єктом може бути складський документ, товарна позиція або пакувальна одиниця. Сканування виконується з використанням вбудованого або зовнішнього сканера штрих-кодів (рис. 3.24).

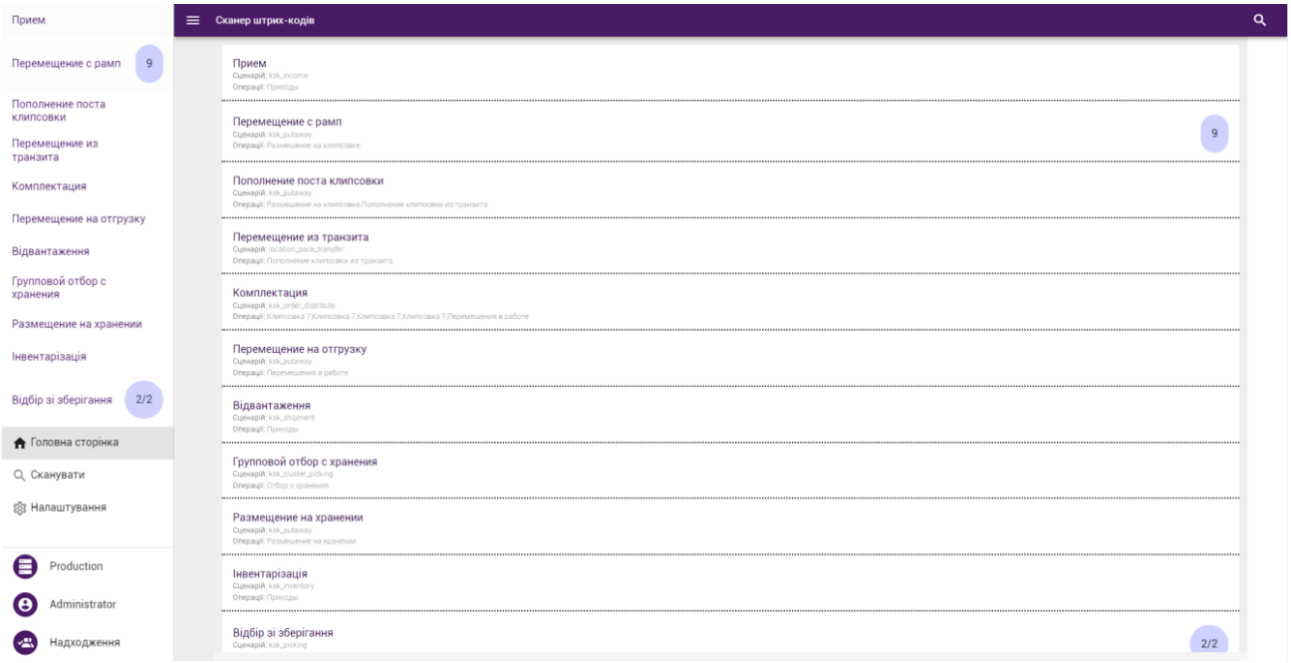


Рисунок 3.23 – Інтерфейс ТЗД

Після зчитування коду система автоматично переходить до наступного екрану, де відображається повний набір доступних дій у межах обраної операції. Наприклад, у випадку приймання товару користувач отримує можливість

переглянути перелік позицій, що підлягають обробці, виконати коригування кількостей, зафіксувати фактичні значення та підтвердити завершення документа (рис. 3.25).

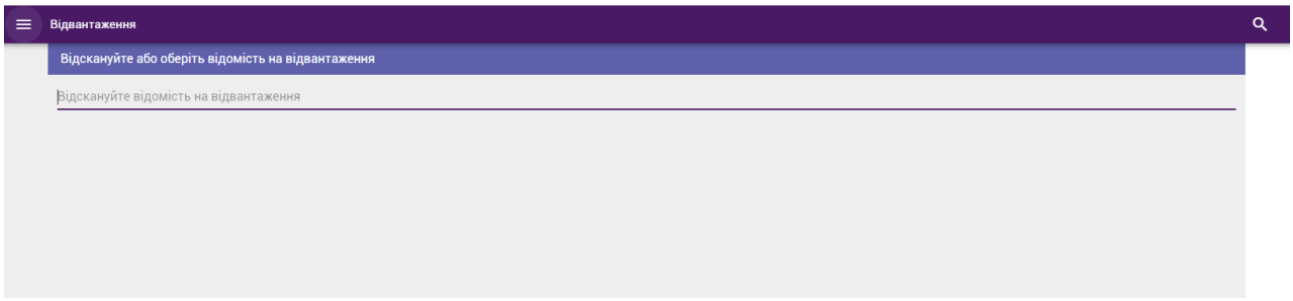


Рисунок 3.24 – Інтерфейс сканування

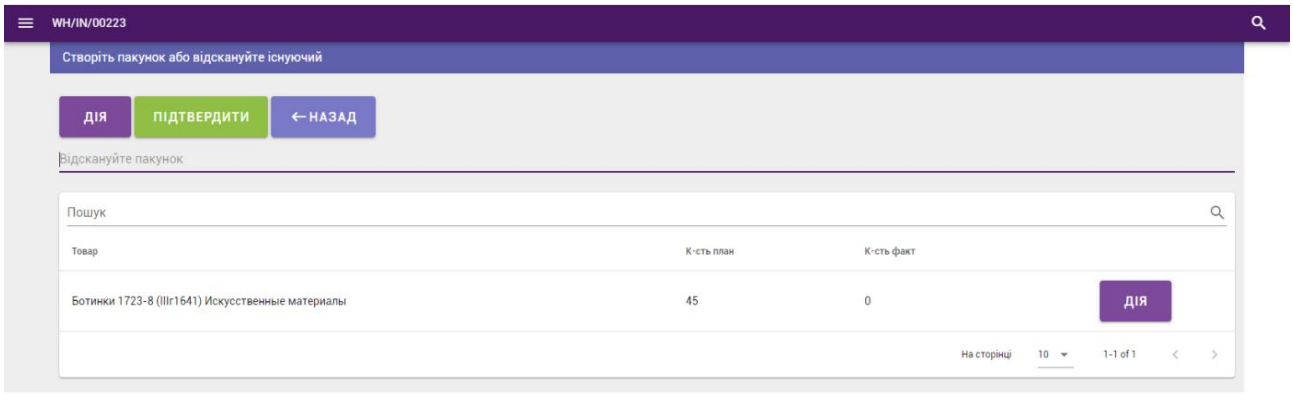


Рисунок 3.25 – Успішне сканування документа

Інтерфейс ТЗД також забезпечує доступ до довідкової інформації щодо товарів, включаючи їх найменування, характеристики, облікову та фактичну кількість на складі, а також інші параметри, необхідні для оперативного прийняття рішень під час виконання складських операцій.

Реалізована логіка інтерфейсу орієнтована на мінімізацію кількості дій оператора, зменшення ймовірності помилок та підвищення швидкості виконання типових складських процесів у режимі реального часу.

3.5.2 Огляд інтерфейсу системи

Ключовим об’єктом у WMS-системі є документ складського переміщення (рис. 3.26). Саме він використовується для реєстрації та контролю переміщення товарно-матеріальних цінностей між різними зонами, комірками або складами. Документ переміщення виступає базовим інструментом управління логістичними процесами та забезпечує актуалізацію даних про запаси в режимі реального часу.

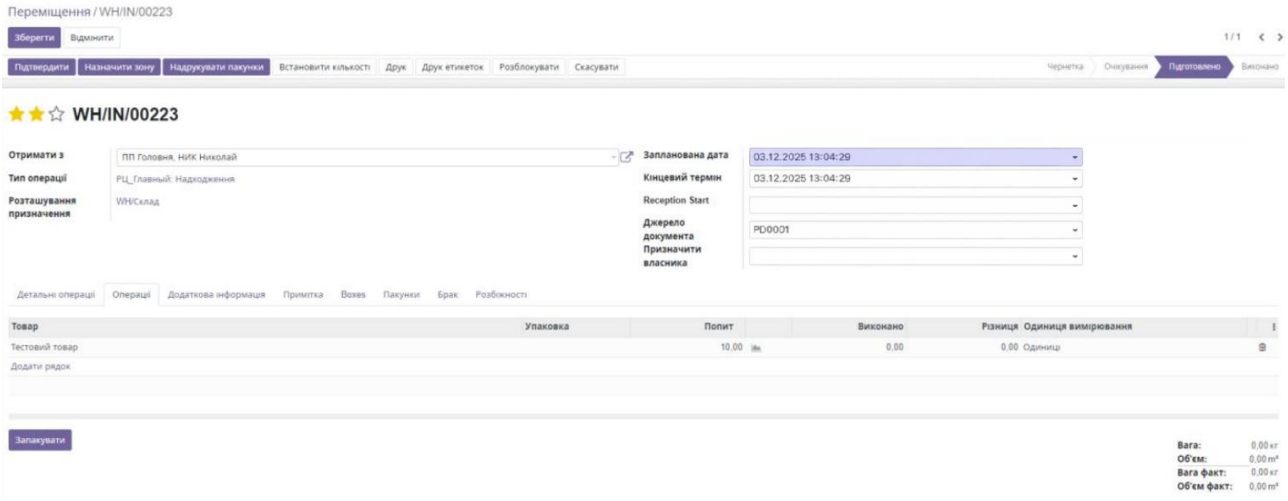


Рисунок 3.26 – Інтерфейс документу переміщення

Формування документа переміщення може здійснюватися як у ручному режимі оператором складу, так і автоматично системою WMS на підставі подій, що виникають у процесі роботи. До таких подій належать приймання нових поставок, відвантаження замовлень клієнтам, внутрішні переміщення для оптимізації зберігання або перерозподілу товарів. Після створення документ переходить у відповідний статус виконання та може бути переданий для опрацювання на робочі місця персоналу або на термінали збору даних.

Важливою складовою інтерфейсу системи є картка товару, яка забезпечує централізоване зберігання та керування інформацією про кожну номенклатурну позицію (рис. 3.27). У картці зосереджені основні та додаткові характеристики

товару, необхідні для ефективного управління запасами, планування складських операцій та оптимізації логістичних процесів.

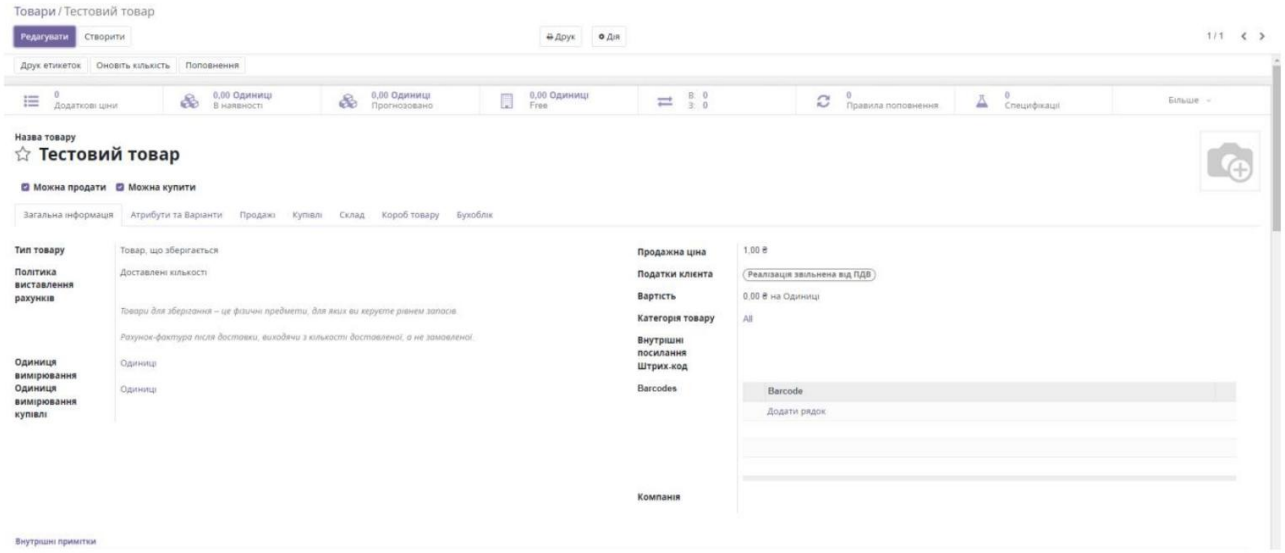


Рисунок 3.27 – Інтерфейс картки товару

Доступ до повної інформації про товар здійснюється через відповідну картку, яка відкривається з переліку номенклатури. Інтерфейс системи дозволяє переглядати всі ключові параметри товару, а за наявності відповідних прав – редагувати їх та доповнювати. Такий підхід забезпечує гнучке налаштування системи під специфіку складу та підвищує точність обліку товарно-матеріальних цінностей.

3.6 Тестування системи

В екосистемі Odoo передбачено вбудований інструментарій автоматизованого тестування – Odoo Testing Framework, який використовується для перевірки коректності роботи програмних модулів на різних рівнях. Даний фреймворк орієнтований на тестування функціональності, інтеграційних взаємодій та стабільності системи в цілому.

Odoo Testing Framework дозволяє формувати автоматизовані тестові сценарії для перевірки бізнес-логіки, поведінки модулів, обробки даних і

взаємодії користувача з інтерфейсом (рис. 3.28). Засоби фреймворку підтримують створення тестових наборів, генерацію вхідних даних, централізований запуск тестів та контроль результатів їх виконання. Це дає змогу швидко виявляти логічні помилки та невідповідності ще на етапі розробки або модифікації модулів.

```
class TestActionsChangePackageLot(CommonCase):
    """Tests covering changing a package on a move line"""

    @classmethod
    def setUpClass(cls):
        super().setUpClass()
        with cls.work_on_actions(cls) as work:
            cls.change_package_lot = work.component(usage="change.package.lot")

    @classmethod
    def setUpClassVars(cls):
        super().setUpClassVars()
        cls.wh = cls.env.ref("stock.warehouse0")
        cls.picking_type = cls.wh.out_type_id

    def _create_picking_with_package_level(self, packages):
        picking_form = Form(self.env["stock.picking"])
        picking_form.partner_id = self.customer
        picking_form.origin = "test"
        picking_form.picking_type_id = self.picking_type
        picking_form.location_id = self.stock_location
        picking_form.location_dest_id = self.packing_location
        for package in packages:
            with picking_form.package_level_ids_details.new() as move:
                move.package_id = package
        picking = picking_form.save()
        picking.action_confirm()
        picking.action_assign()
        return picking
```

Рисунок 3.28 – Приклад тест-кейсу

Функціональні можливості фреймворку охоплюють перевірку очікуваних результатів, керування тестовими станами, перехоплення виключень та моделювання типових і нештатних сценаріїв роботи системи. Тестування може виконуватися як для окремих модулів, так і для всієї системи, що забезпечує комплексну оцінку працездатності WMS-рішення.

Застосування Odoo Testing Framework дозволяє суттєво підвищити надійність і стабільність програмного забезпечення, зменшити ризик

виникнення критичних помилок під час розгортання нових функцій або оновлення системи. Регулярне використання автоматизованого тестування сприяє підвищенню якості розробки та довіри користувачів до WMS-системи.

Запуск тестування здійснюється шляхом старту локального сервера Odoo з додатковими параметрами, зокрема із зазначенням бази даних, модуля та увімкненням режиму тестування, наприклад: *-d {назва_бази_даних} -i {назва_модуля} --test-enable*.

Для перевірки коректності роботи інтерфейсу ТЗД було застосовано сценарій із навмисним введенням помилкових даних (рис. 3.29 – 3.30). Такий підхід дозволив переконатися, що система коректно обробляє невалідні запити, не знаходить неіснуючі документи та не призводить до зупинки або порушення логіки роботи WMS.

Рисунок 3.29 – Невдалий результат тестування

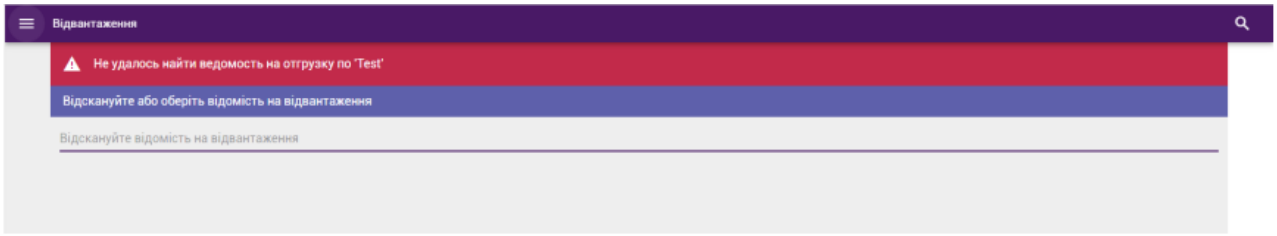


Рисунок 3.30 – Невдалий результат сканування

Успішне проходження тестування свідчить про стабільну роботу WMS-системи на базі Odoo, відповідність реалізованого функціоналу вимогам складської логістики та дотримання основних принципів надійності й масштабованості сучасних WMS-рішень.

3.7 Висновки до розділу

У третьому розділі виконано практичну реалізацію системи автоматизації логістичного центру виробничого підприємства на базі ERP-платформи Odoo. На початковому етапі було розгорнуто та налаштовано середовище розробки, підготовлено структуру користувацького модуля та забезпечено його інтеграцію зі стандартною підсистемою складського обліку (Inventory). Реалізація здійснювалася відповідно до проєктних рішень, сформованих у попередньому розділі.

Було розроблено серверну частину WMS-підсистеми з використанням об’єктно-орієнтованого підходу. Реалізовано розширення стандартних моделей Odoo (StockPicking, StockMove, StockMoveLine) та впроваджено додаткову модель Divergence для автоматизованого виявлення і фіксації розбіжностей між плановими та фактичними параметрами складських операцій. Це дозволило забезпечити контроль коректності виконання операцій у режимі реального часу та підвищити достовірність обліку товарно-матеріальних цінностей.

Розроблено та налаштовано елементи користувацького інтерфейсу, що забезпечують зручну взаємодію персоналу складу з системою. Реалізовано підтримку роботи з терміналами збору даних, що дало змогу автоматизувати процес введення інформації під час виконання складських операцій і мінімізувати вплив людського фактору. Інтерфейсні рішення забезпечують оперативний доступ до інформації про стан складських процесів і виявлені розбіжності.

Проведене тестування підтвердило працездатність розробленої системи та коректність реалізованих алгоритмів. Система стабільно обробляє типові складські операції (приймання, відвантаження, переміщення товарів), своєчасно оновлює дані про складські залишки та забезпечує фіксацію відхилень у процесі виконання операцій. Отримані результати свідчать про відповідність розробленої WMS-підсистеми поставленим вимогам та її готовність до практичного використання в логістичному центрі підприємства.

4 ОХОРОНА ПРАЦІ

Сучасний етап розвитку промисловості та сфери послуг характеризується безперервним впровадженням засобів автоматизації й цифровізації, що суттєво змінює характер виробничих і управлінських процесів. Функціонування більшості підприємств у теперішніх умовах є неможливим без широкого застосування комп'ютерної техніки та інформаційних систем, які забезпечують обробку, зберігання та передачу даних.

З огляду на масове використання комп'ютерів у професійній діяльності працівників, в Україні діє чітко визначена нормативно-правова база, яка регламентує умови безпечної експлуатації електронно-обчислювальної техніки. Зокрема, вимоги до організації праці встановлені Правилами охорони праці під час експлуатації електронно-обчислювальних машин, затвердженими наказом уповноваженого державного органу у 2010 році. Зазначені правила поширюються на всі суб'єкти господарювання незалежно від форми власності, якщо їх діяльність пов'язана з використанням електронно-обчислювальних машин, відеодисплейних терміналів та периферійного обладнання. Нормативний документ визначає обов'язкові вимоги до безпеки робочих місць операторів і порядку виконання робіт із застосуванням комп'ютерної техніки.

Крім того, санітарно-гігієнічні умови праці регламентуються Державними санітарними правилами і нормами ДСанПіН 3.3.2.007-98, які встановлюють допустимі параметри організації праці при роботі з візуальними дисплейними терміналами. Дані норми охоплюють вимоги до мікроклімату, освітлення, рівнів шуму, вібрації та впливу фізичних полів для персональних і колективних електронно-обчислювальних систем незалежно від країни їх виробництва.

Приміщення, у яких передбачається встановлення комп'ютерного обладнання, повинні відповідати затвердженій проєктній документації будівлі та бути погодженими з відповідними контролюючими органами. Роботодавець зобов'язаний забезпечити дотримання нормативів освітленості, параметрів

температури та вологості повітря, допустимих рівнів шуму, вібрації, а також безпечних значень електромагнітного, ультрафіолетового та інфрачервоного випромінювання. Конкретні допустимі показники зазначених параметрів визначені у чинних санітарних нормах.

Нормативними документами прямо забороняється розміщення комп'ютерних робочих місць у підвальних та цокольних приміщеннях. Приміщення з відеодисплейними терміналами мають бути оснащені системами опалення, вентиляції або кондиціонування повітря. При цьому всі інженерні комунікації (труби, радіатори, вентиляційні канали) повинні бути ізольовані або захищені спеціальними кожухами для запобігання ураженню електричним струмом.

Робочі приміщення для персоналу, який працює з комп'ютером, повинні мати як природне, так і штучне освітлення. Для регулювання світлового потоку на вікнах необхідно встановлювати жалюзі або штори. Робочі місця слід орієнтувати таким чином, щоб денне світло надходило збоку, переважно з лівого боку. Відстані між відеодисплейними терміналами повинні відповідати ергономічним вимогам: між бічними поверхнями екранів – не менше 1,2 м, між тильною поверхнею одного монітора та екраном іншого – не менше 2,5 м.

Робочі столи мають відповідати ергономічним стандартам і забезпечувати раціональне розміщення комп'ютерного обладнання та робочої документації. Робочі крісла повинні бути поворотними, регульованими за висотою та кутами нахилу, із напівм'яким покриттям, що не накопичує статичну електрику та легко очищується. Мінімальна площа одного робочого місця повинна становити не менше 6 м², а об'єм повітря – не менше 20 м³ .

Допускається розміщення на робочому столі допоміжних пристроїв (принтерів, сканерів, акустичних систем) за умови, що це не погіршує огляд екрана та не створює додаткових незручностей для працівника. У разі виникнення підвищеного шуму або вібрацій роботодавець зобов'язаний забезпечити використання спеціальних антивібраційних засобів [19].

Санітарний стан приміщень повинен підтримуватися шляхом щоденного вологого прибирання, а також регулярного очищення робочих поверхонь і моніторів від пилу.

На підприємстві забороняється:

– виконувати ремонт або технічне обслуговування комп’ютерної техніки безпосередньо на робочому місці;

– самостійно здійснювати ремонт або налаштування обладнання без участі кваліфікованих фахівців;

– захаращувати робоче місце сторонніми предметами та документацією;

– експлуатувати комп’ютери з ознаками несправностей, нестабільним зображенням або нехарактерними сигналами;

– відключати захисні пристрої та вносити несанкціоновані зміни до конструкції обладнання;

– допускати до роботи осіб, які не пройшли обов’язковий інструктаж і перевірку знань з охорони праці та пожежної безпеки.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було комплексно досліджено, спроєктовано та реалізовано систему автоматизації логістичного центру виробничого підприємства на основі ERP-платформи Odoo з розширенням її функціональності до рівня Warehouse Management System (WMS).

У першому розділі було проаналізовано сучасний стан і тенденції розвитку систем автоматизації складської логістики. Досліджено базові концепції ERP та WMS, а також розглянуто функціональні можливості комерційних і відкритих програмних рішень для управління складом. У результаті аналізу встановлено, що традиційні методи управління складськими процесами не забезпечують належного рівня оперативності, точності та прозорості обліку. Обґрунтовано доцільність використання WMS-підходу для автоматизації логістичного центру та доведено, що платформа Odoo є раціональним вибором завдяки модульній архітектурі, відкритому вихідному коду та можливості інтеграції складської логістики з іншими бізнес-процесами підприємства.

У другому розділі було здійснено обґрунтування вибору технологій та середовища розробки, а також виконано проектування системи. Розроблено структурну схему, що відображає трирівневу архітектуру рішення (рівень бази даних, рівень бізнес-логіки та рівень користувацького інтерфейсу), і функціональну модель у вигляді діаграми класів, яка формалізує основні сутності складської підсистеми та взаємозв’язки між ними. Крім того, сформовано алгоритм роботи програмного забезпечення, що описує послідовність виконання складських операцій із використанням терміналів збору даних та механізму контролю розбіжностей. Результати другого розділу створили логічну та методичну основу для подальшої програмної реалізації системи.

У третьому розділі виконано практичну реалізацію розроблених проектних рішень. Було створено користувацький модуль Odoo, який розширює

стандартний функціонал складського обліку та забезпечує автоматизоване управління складськими операціями. Реалізовано серверну логіку з модифікацією базових моделей Odoo та впровадженням механізму фіксації розбіжностей між плановими і фактичними параметрами операцій. Налаштовано користувацький інтерфейс та підтримку роботи з терміналами збору даних, що дозволило зменшити кількість ручних операцій і підвищити достовірність облікових даних. Проведене тестування підтвердило працездатність системи, коректність реалізованих алгоритмів та її відповідність вимогам до автоматизації логістичного центру.

У процесі розробки системи було виявлено низку проблемних аспектів:

– виявлено підвищену складність адаптації стандартних моделей платформи Odoo до специфічних вимог складської логістики, що зумовило необхідність детального аналізу внутрішньої архітектури системи та обережного розширення її функціоналу без порушення базової логіки;

– встановлено, що інтеграція терміналів збору даних потребує додаткової валідації вхідних даних для запобігання дублюванню або некоректній обробці результатів сканування, особливо за умов інтенсивного виконання складських операцій;

– при масштабуванні системи (збільшенні кількості користувачів та обсягу операцій) можуть виникати підвищені вимоги до продуктивності серверної частини та оптимізації доступу до бази даних.

Подальший розвиток розробленої системи автоматизації логістичного центру доцільно здійснювати за такими напрямами:

– впровадження технологій RFID та елементів Інтернету речей (IoT) для автоматизованої ідентифікації товарів і моніторингу їх переміщення без участі оператора, що підвищить рівень автоматизації складських процесів;

– розширення аналітичного функціоналу системи шляхом використання методів прогнозування та аналізу даних для оптимізації складських запасів і підтримки прийняття управлінських рішень;

Таким чином, завдання кваліфікаційної роботи вирішено. Мета роботи, що полягала у підвищенні ефективності керування складськими процесами за рахунок створення системи автоматизації, яка забезпечує раціональну організацію потоків матеріальних ресурсів, мінімізацію впливу людського фактора та підвищення точності виконання логістичних операцій, досягнута.

Отже, розроблена система є ефективним інструментом оптимізації складської логістики та створює передумови для подальшого розвитку цифрової інфраструктури підприємства.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Положення про організацію освітнього процесу у ХНУРЕ [електронний ресурс]: https://nure.ua/wp-content/uploads/Main_Docs_NURE/polozhennja-proorganizaciju-osvitnogo-procesu-v-hnure-2023.pdf (дата звернення: 30.10.2025).

2. Положення про академічну доброчесність [Електронний ресурс]: Наказ ХНУРЕ від 02 лютого 2021 р. No50. – Режим доступу: https://nure.ua/wpcontent/uploads/Main_Docs_NURE/polozhennja-pro-akademichnu-dobrochesnist.pdf (дата звернення: 30.10.2025).

3. Стандарт вищої освіти за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології» галузі знань 15 «Автоматизація та приладобудування» для другого (магістерського) рівня вищої освіти, затверджений наказом МОН України No 1022 від 10.08.2020р. «Про затвердження стандарту вищої освіти за спеціальністю 151 Автоматизація та комп'ютерно-інтегровані технології для другого (магістерського) рівня вищої освіти» Режим доступу: https://mon.gov.ua/static-objects/mon/sites/1/vishchaosvita/zatverdzeni%20standarty/2020/08/10/151-avtomatizatsiya-ta-kit-magistr.pdf (дата звернення: 30.10.2025).

4. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітньо-41 професійних програм: «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2024. 57 с.

5. Основи наукових досліджень: підручник / І. Ш. Невлюдов, Ю. М. Олександров, А. О. Андрусевич, О. О. Чала; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Prague: OKTAN PRINT, 2024. – 468 с. DOI

https://doi.org/10.46489/ONDNP Режим доступу на ресурсі бібліотеки ХНУРЕ https://openarchive.nure.ua/handle/document/28574 (дата звернення: 30.10.2025).

6. Невлюдов І. Ш. Техніко-економічне обґрунтування інженерних рішень в інтелектуальному виробництві: підручник / І. Ш. Невлюдов. - Кривий Ріг: Чернявський Д. О., 2024. – 388 с.: іл. Режим доступу на ресурсі бібліотеки ХНУРЕ https://openarchive.nure.ua/handle/document/27408

7. Невлюдов І.Ш. Автоматизована система керування технологічними процесами в SCADA системі TRACE MODE 6: Навчальний посібник / І.Ш.Невлюдов, А.О. Андрусевич, В.В. Євсєєв, С.С. Максимова, М.Г. Стародубцев, В.В.Невлюдова. – Кривий Ріг: Криворізький коледж НАУ, 2018. – 320 с.

8. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання / Нац. стандарт України. – Вид. офіц. – [Чинний від 2017 – 07 – 01]. – Київ: ДП «УкрНДНЦ», 2016. – 26 с.

9. Невлюдов, І., Слюсар, А., Хрустальова, С., Хрустальов, К., & Косенко, В. (2023, November). Порівняння ефективності застосування технологій штрихового кодування та RFID у логістичних процесах. In 2023 2nd International Conference on Innovative Solutions in Software Engineering (ICISSE) (p. 183).

10. Ткаченко І. Автоматизація логістичних процесів виробничого підприємства / І. Ткаченко // Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2025) : збірник студентських наукових статей, 2025. – Харків : ХНУРЕ, 2025. – Вип. 1. – С. 132-136.

11. Крюковська С., Жарська І. Проблеми та перспективи автоматизації складів у транспортній логістиці. Економіка та суспільство. 2024. № 68. URL: https://doi.org/10.32782/2524-0072/2024-68-99 (дата звернення: 15.10.2025).

12. Logistics 4.0 in warehousing: a conceptual framework of influencing factors, benefits and barriers / S. Perotti та ін. The international journal of logistics management. 2022. URL: https://doi.org/10.1108/ijlm-02-2022-0068 (дата звернення: 15.10.2025).

13. Perkumienė D., Vienažindienė M. Innovative solutions in the warehousing processes of manufacturing companies towards sustainability. Journal of infrastructure, policy and development. 2024. Т. 8, № 8. С. 7084. URL: https://doi.org/10.24294/jipd.v8i8.7084 (дата звернення: 16.10.2025).

14. Tubis A. A., Rohman J. Intelligent warehouse in industry 4.0–systematic literature review. *Sensors*. 2023. Т. 23, № 8. С. 4105. URL: https://doi.org/10.3390/s23084105 (дата звернення: 16.10.2025).

15. Корман І., Семенда О., Мазур Ю. Вплив цифрових технологій на управління каналами розподілу та логістику в умовах глобальної економіки. Економіка та суспільство. 2025. № 71. URL: https://doi.org/10.32782/2524-0072/2025-71-28 (дата звернення: 20.10.2025).

16. Jin Y. L., Gong A. L., Liu S. P. Study of logistics warehouse management system based on RFID. Advanced materials research. 2013. Т. 748. С. 1281–1284. URL: https://doi.org/10.4028/www.scientific.net/amr.748.1281 (дата звернення: 20.10.2025).

17. Tikwayo L. N., Mathaba T. N. D. Applications of industry 4.0 technologies in warehouse management: a systematic literature review. Logistics. 2023. Т. 7, № 2. С. 24. URL: https://doi.org/10.3390/logistics7020024 (дата звернення: 21.10.2025).

18. Zhang Y. Smart logistics warehouse management system based on RFID and internet of things technology. 2022 2nd international conference on networking, communications and information technology (netcit), м. Manchester, United Kingdom, 26–27 груд. 2022 р. 2022. URL: https://doi.org/10.1109/netcit57419.2022.00121 (дата звернення: 21.10.2025).

19. Закон України «Про охорону праці» // Відомості Верховної Ради України (ВВР). 1992. с.668. URL: http://zakon4.rada.gov.ua/laws/show/2694-12. (дата звернення 03.12.2022).